FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

A Secure and Privacy Preserving Approach to Medical Data Mining Applications

Gustavo de Castro Nogueira Pinto

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: José Manuel de Magalhães Cruz

Co-Supervisor: Rui Carlos Camacho de Sousa Ferreira da Silva

July 23, 2018

A Secure and Privacy Preserving Approach to Medical Data Mining Applications

Gustavo de Castro Nogueira Pinto

Mestrado Integrado em Engenharia Informática e Computação

July 23, 2018

Abstract

As information systems proliferate in medical institutions, the volume of medical data and the need for its secure analysis and sharing increases, such as for data mining purposes.

Medical data often carries sensitive and personal information that lead to two points of failure on the assurance of a secure and private transmission. Firstly, portions of data that can, alone or crossed with other data, identify the patient; secondly, the transmission itself of data between two remote devices.

Infrastructure setups and mechanisms are thus proposed to ensure proper data anonymization, while satisfying proper risk thresholds and minimizing data loss. Furthermore, the infrastructure also allows for a secure transmission of data, including authentication of the parties involved, integrity of the exchanged messages and privacy of the communications.

Finally, the usage of incremental data mining algorithms is emerging as a strong method to manage large quantities of data in a distributed fashion. Achieving implementations of incremental data mining algorithms that use anonymized and secured data and can still perform competitively with standard data mining techniques is also an addressed concern.

The solution will benefit researchers or institutions that handle medical data with multiple data mining systems in different locations. Alternatively, providers of medical data can also explore options to safely transmit and to correctly manage the risks of de-anonymization of their datasets.

Resumo

À medida que os sistemas de informação proliferam em instituições médicas, o volume de dados médicos e a necessidade das suas análise e partilha seguras aumentam, como para propósitos de data mining.

Dados médicos frequentemente carregam informação pessoal e sensível que podem levar a dois pontos de falha de uma transmissão segura e privada. Primeiramente, porções dos dados que podem, sozinhas ou com auxílio de outros dados, identificar um paciente; em segundo lugar, a transmissão em si dos dados entre dois dispositivos remotos.

Configurações de infraestrutura e outros mecanismos são, então, propostos de modo a garantir anonimização correta de dados, satisfazendo limites aceitáveis de risco de re-identificação e tentado minimizar a perda de informação. Além disso, a infraestrutura permite também uma transmissão segura de dados, incluíndo autenticação das entidades envolvidas, integridade das mensagens trocadas e privacidade das comunicações.

Finalmente, o uso de algoritmos de data mining incrementais emergem como um bom método capaz de lidar com grandes quantidades de dados, num contexto distribuído. Um dos objetivos é, também, alcançar implementações de algoritmos de data mining incrementais que utilizem dados anonimizados e seguros e que consigam, ainda, ser competitivos com técnicas de data mining tradicionais.

A solução apresentada beneficiará investigadores ou instituições que lidem com dados médicos com múltiplos sistemas de data mining em localizações diferentes. Alternativamente, provedores de dados médicos podem explorar opções para, seguramente, transmitir e corretamente gerir os riscos de re-identificação dos seus conjuntos de dados.

Acknowledgements

This space is dedicated exclusively to the plethora of people who have surrounded and directly and indirectly supported the development of this work.

Firstly, it is indispensable that I mention both my supervisors, whom I hold in a very high regard. Professor José Magalhães Cruz for the availability and immaculate diligence to guide, correct and push this work into higher standards, and Professor Rui Camacho for all the high-spirited meetings and messages exchanged.

My family, who has been the corner stone of all my achievements.

And all the friends that I hold dear, with whom I have shared countless hours and days and nights, reminiscing on what was done, talking about what is being done and dreaming about what is to be done.

Gustavo de Castro Nogueira Pinto

"Odi et amo. quare id faciam fortasse requiris.
nescio, sed fieri sentio et excrucior"

Catullus

Contents

List of Figures

List of Tables

Abbreviations

BEAST	Browser Exploit Against SSL/TLS
CA	Certificate Authority
CRIME	Compression Ratio Info-leak Made Easy
CRISP-DM	CRoss Industry Standard Process for Data Mining
CSR	Certificate Signing Request
DM	Data Mining
EHR	Electronic Health Record
FQDN	Fully-Qualified Domain Name
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
LM	Loss Metric
MAC	Message Authentication Code
NTP	Network Time Protocol
ORF	Online Random Forest
OSSTMM	Open Source Security Testing Methodology Manual
PKI	Public Key Infrastructure
RFC	Request For Comments
SSL	Secure Socket Layer
SVM	Support Vector Machine
TLS	Transport Layer Security
VLBW	Very Low Birth Weight
VPT	Very PreTerm

Chapter 1

Introduction

In current days information is being created in amounts never seen before, quantities that can swiftly overshadow the information left on records throughout centuries. Specifically, medical data follows the same trend with the ongoing developments on integrating automation into medical practices allied with the growing accessibility to health care. Access to large sets of data brings a potential to extract more information from them, such as identifying patterns and relations between elements. Data Mining is an area that attempts to identify all kinds of interesting meta information and apply them to useful processes, such as attempting to predict outcomes of future situations.

Nonetheless, even with the assistance of machines, the resources on a single location may be scarce to successfully match the processing rate of the data with its creation rate. Therefore, distributed data mining systems may be applied, where different locations or entities join efforts to attempt to harmoniously extract useful information from the data.

Alternatively, different entities, located in different places and with different resources may want to access the same information or share results or ongoing processes between their partners, even if the ultimate goal of the entities does not align. This presents another scenario where a distributed data mining setting can exist.

Scenarios with information transfers happening over a plural number of entities brings some different concerns with special relevance on the confidentiality and privacy issues, magnified when the transfered information is medical data which is considered to be highly sensitive, many times with laws dictating the security assurances that must be applied to this kind of data, as well as the liability in the case of any malpractice [MT14]. As a consequence, proper security implementations in a distributed setting are pivotal for a law abiding and responsible development.

1.1 Context

At this date there is an ongoing project on an European scale that attempts to develop a platform for conducting multiple operations, structuring and collecting data from children and adults born very

preterm (VPT) or born with a very low birth weight (VLBW). The information will be gathered from various different national and European cohorts of VPT/VLBW births.

Globally, the aim is improving the overall health of individuals who were born VPT/VLBW by developing a platform that aggregates relevant information which is very geographically diverse and not trivial to obtain (VPT/VLBW births account for less than 2% of births across Europe). Also, the platform must contain solid mechanisms that can facilitate access and transmission of this data to different kinds of research, such as in the health care area, within legal boundaries.

The European project is divided into different work plans, ranging from matters such as legal implications of the governance of this sort of data, up to development of a data platform with correct web portals and accesses, and going through statistical methods (such as some used in data mining) to analyze the data that was collected and demonstrate its usefulness.

Consequently, the solution here described comes as an approach to the transmission of data between the different parties involved, providing a prototypical approach to the correct anonymization of data for a (usually remote) requester without access to the original dataset, or an approach that does not share sensitive data at all, and demonstrating preserved value to data mining methods that can be applied in this distributed setting.

1.2 Objectives

The goal of the dissertation work is the design and prototyping of a secure architecture containing distributed data mining applications and different data sources. To expand, this distributed architecture must contain processes able to assure three different aspects:

- **Medical data anonymization:** Under some established risk, the data must be properly anonymized in a way that impedes occurrence of re-identification, while still attempting to minimize information loss. For example, an anonymized Electronic Health Record (EHR) should not be able to be re-identified to the individual it corresponds to. Alternatively, the medical datasets must not be unduly shared and only be applied locally, by the entity who has the correct permissions to hold them.

- **Secure data transfer:** When transferring information between points or entities, the transfer must be protected against different attack vectors or vulnerabilities, preventing unauthorized access or corruption of the information in transit.

- **Useful data mining applications:** Even with a distributed setting and properly anonymized sets of information, the data mining applications in the architecture shall remain with a considerable degree of utility by still being able to extract useful meta information.

The evaluation of the resulting architecture must be divided in area specific metrics, in an attempt to evaluate the results on each of the tasks the architecture encompasses, i.e.:

- Data anonymization processes will measure the information that was lost to achieve scientifically consensual re-identification risk thresholds for medical data, in an example dataset.

- The security of the data transfers shall be evaluated by implementing state of the art mechanisms and by performing security testing following standard and proven guidelines and including the assistance of well established tools.
- Different Data Mining processes utilize different methods for its own performance evaluation and the same metrics will be used to perform comparisons between a local process with distributed incremental processes, while utilizing anonymized and non-anonymized data.

The last item describes a metric that may be used as a measure of performance for the whole architecture, since it evaluates the performance of data mining application in a standard local environment against data mining applications after undergoing all the processes that the architecture enforces on the data.

1.3 Dissertation Structure

This introductory chapter is followed by chapter 2, where state of the art mechanisms for the different areas involved in the dissertation are presented.

A third chapter will introduce an overview of the solution, including different use cases and how different components of the developed work interact.

Chapter 4 describes methods used to correctly anonymize a medical dataset and includes some empirical information about the achieved result.

Following, another chapter presents the utilized methods relative to data transmission security and experimental setups to assure a successful implementation.

Throughout chapter 6, all data mining mechanisms implemented and their performances are evaluated and compared within the different inputs and scenarios they were applied in.

Conclusions and future work will be presented in a final chapter.

Chapter 2

Bibliographic Review

2.1 Introduction

This chapter is divided in three sections that represent the three main areas involved in the project, data anonymization, data transfer security and data mining, and a final one addressing incremental machine learning.

2.2 Data Anonymization

Data anonymization attempts to protect data in a way that if anyone has access to the data, the privacy of the information is protected. In other words, a record in a dataset should not be able to be traced back (re-identified) to the original individual or entity to which it belongs to, under some acceptable risk threshold.

To accomplish this, different types of variables in a dataset have to be identified and studied and methods to anonymize a specific dataset, while attempting to minimize data loss, have to be applied.

2.2.1 Types of attributes

In a medical dataset there will be different attributes that will be studied and possibly operated over different processes of data anonymization.

Although [EE11] specifies a rather detailed taxonomy for the classification of the attributes, in a more general sense they can be divided in just two groups of variables:

- **Direct Identifiers:** These consist of attributes who can single-handedly identify the patient or individual to whom, for instance, an Electronic Health Record (EHR) belongs to. This information is extremely sensitive and these identifiers rarely are unaltered by data anonymization techniques. Even in a situation where a direct identifier occurs multiple

times in a dataset, its unaltered existence makes it very dangerous for other information to be included, because it can be cross-referenced with other attributes to identify the individual. As an example, even if there are plenty of records with the same name in the dataset, matching the name with an area, or an e-mail address, can present a big re-identification risk. Other examples of direct identifier would be a social identification number or a credit card number, which are unique and correspond to a single individual.

- **Indirect Identifiers:** Unlike the direct identifiers, indirect identifiers (sometimes called quasi-identifiers) alone would not be able to directly identify an individual. Nonetheless, when cross-referenced with other identifiers or in a specific dataset they can be able to do so probabilistically. For instance, the age of a patient is usually harmless, but if the dataset of an hospital has only one patient who is over 100 years old, the age attribute could be enough to determine the individual to whom the EHR corresponds to. The same logic can be applied to a set of indirect identifiers instead of a single one: if there is only one record of a 30 year old female from a specific area who was admitted in the hospital in a particular day, those 4 identifiers (age, gender, day of admission and area) in conjunction can re-identify the patient.

2.2.2 Standard de-identification techniques

[EE11] documents the existence of three major de-identification methods, each of which can be applied in different situations.

- **Masking:** this method is applied to direct identifiers and the underlying techniques can vary. Depending on the dataset, on the direct identifier and on the context of the application, (reversible) encryption may be applied so information can — with a proper request to the the entity that performed the masking — be decrypted in an authorized re-identification situation (for example in a judicial case). Other techniques such as pseudonymization can be used, where direct identifiers are replaced by irreversible unique numbers or strings.

- **Generalization:** a method used upon indirect identifiers. One of the most common techniques, named *hierarchy-based generalization*, consists of predefining a hierarchy with an increasingly broad range of an attribute and the hierarchy will iteratively be climbed until re-identification risk thresholds are achieved. Table 2.1 contains an example of a predefined hierarchy for a date that could be something such as an admission date on a hospital.

- **Supression:** used on records that are marked for deletion, often discovered while evaluating the dataset for masking. Supression may be executed in different ways, going as broad as removing the whole record while sometimes removing a single attribute from a specific row can suffice.

Precision	Example
Day	10 Jul 2016
Month	Jul 2016
Year	2016
Decade	2010-2019

Table 2.1: Hierarchy-based generalization.

2.2.3 Risk Assessment

2.2.3.1 Attack Types

As documented by [EED08], an attack on a dataset may fall into two different categories. The first one is named the *prosecutor scenario* and the second one being the *journalist scenario*.

On the prosecutor scenario, an attacker will search the dataset for a particular record. So, in a EHR dataset, the prosecutor will know the individual he is looking for and try to recognize distinguishing features or identifiers (see 2.2.1), that may identify the EHR as belonging to the targeted individual.

Differently, the journalist scenario will indiscriminately try to re-identify any individual in the dataset. Instead of searching for a particular individual, an attacker in a journalist scenario may have access to some demographic information about all individuals in an area and try to match that information he possesses with the dataset, singling out as many individuals as possible.

2.2.3.2 K-Anonymity

K-anonymity was originally proposed by [SS98] and consists of a property that a dataset may contain. It dictates the minimum number (k) of records in a dataset that have similar indirect identifiers, for any set of indirect identifiers.

Both [SS98] and [EED08] describe approaches to enforcing this property on a dataset and they mostly rely on the techniques described in 2.2.2, with special attention to the attempt to minimize the information lost in these processes. Also with this data loss minimization goal in mind, some optimization algorithms have been developed to attempt to solve the problem, such as [KT12].

2.2.3.3 Risk quantification

A set that shares the same values for any combination of indirect identifiers is named *equivalence class*. Thus, the size of an equivalence class will be determined by the number of records that share the same values for a set of indirect identifiers.

Relating with k-anonymity, the size of the smallest equivalence class of a dataset with a correctly enforced or natural k-anonymity will be k.

With the concept of equivalence class and as proposed by [EED08], the risk associated with the re-identification of a dataset can be calculated for the two different attack scenarios described in 2.2.3.1.

On a prosecutor scenario, since the attacker will be searching for a particular individual, the risk will be $\frac{1}{min(s_e)}$, where $min(s_e)$ is the size of the smallest equivalence class to whom the individual belongs to.

Meanwhile, on a journalist scenario, the risk quantification logic is very similar but since the attacker is looking to re-identify any individual in the database, the equivalence classes with the lowest size will be the most vulnerable targets. The risk will be given by the expression $\frac{1}{min(s)}$, where $min(s)$ is the size of the smallest class in the whole dataset.

2.3 Data Transfer Security

2.3.1 Symmetric and Asymmetric Cryptography

There have been well documented approaches to secure communication mechanisms which vary in complexity, in system setting and in limitations. [KGNK16] presents some methods on how to handle authentication and part of the communication in large systems with many machines involved by describing important concepts used in authentication and communication mechanisms.

Among them exists **public key encryption**, which allows two parties with no prior agreement to safely communicate with each other. [Hel02] explained how the inability for this to happen was a big hindrance in the development of many systems and documents the mathematical foundations of these types of algorithms.

Public key encryption (sometimes also called asymmetric cryptography) mechanisms generate two different keys, a public key and a private key that operate as a pair. Each one of the keys of the pair can encrypt a message that only the other key can decrypt. The role of each key will depend on the type of protective usage. A public key can be freely distributed and obtained by anyone. On the other hand, the private key must be kept a secret that only the generator of the pair can access. This is the main assumption of the whole mechanism.

With this knowledge, and under that assumption, there are different cases of use of these keys:

- When a message is encrypted with a public key of an entity A, it is guaranteed that only A can decrypt the message, for only A possesses the required private key to do so.

- When a message is encrypted with entity A's private key, it can serve as a proof that A was who wrote the message. To confirm this, the message must be attempted to be decrypted by A's public key. If successfully done, A's authorship of the message can be confirmed. Using

this to encrypt what is usually a hash[1] of the message being sent (for performance reasons) is the basis of a **digital signature** [CGI04].

There are other well established types of encryption techniques which are **symmetric key encryption** techniques. In [KS14] it is provided an overview of these techniques which consist of a shared secret between two or more entities, that can use that shared secret (shared key) to communicate securely. With a secure algorithm, it is very highly assured that only someone with access to the key can decrypt the message.

Two noteworthy aspects are that sharing the symmetric shared key over a network between two entities with no prior arrangement can pose to be a problem and that the computational demand of asymmetric cryptography techniques is usually considerably higher compared to the demands of symmetric techniques [PNNM16].

One way to approach both problems is to to utilize public key encryption techniques to share symmetric keys and then the remainder of the communication will be performed using symmetric key methods.

2.3.2 Digital Certificates

Digital certificates consist of digital records that can be transmitted as a proof of an entity's own identity. Therefore, obtaining a digital certificate is pivotal for many systems, with special relevance to entities that have to ensure their identity to their communication partners, such as banking websites. At the core of the whole digital certificate infrastructure there are *Certificate Authorities* (CA) which are entities that are trusted by all the parties in the communication.

To obtain a digital certificate, an entity has to request it to a CA. After verifying the identity of the requester, if successful, the CA will generate a digitally signed certificate (see digital signature in 2.3.1) that contains the requester's public key and other supplementing and administrative information such as serial number, validity period and requester's name.

Therefore, on the core assumption that the CAs are trusted by all the parties involved, a digital certificate can be used to prove an entity's identity, through an ensuing authentication protocol (i.e. challenge-response type). If the signature of the certificate can successfully be verified as said in 2.3.1.

Information in this section about digital certificates was extracted from [Pit03].

There are some standard formats for digital certificates, being one of the most common ones the X509v3 standard, defined in RFC 5280 [HFPS08].

[1] A hash in this context is the result of a cryptographic hashing function, where an input is mapped into fixed size data (the hash). A good hashing function will feature some properties, such as determinism, which states that the same input for the same hashing function must always output the same hash. Furthermore, a cryptographic hashing function must also guarantee low collision probability — high unlikeliness that different inputs result in the same hash. Lastly, non-reversibility — a property that states that one cannot obtain the input from its hash. [AA13]

2.3.3 Data Integrity

Prevention of any data corruption, especially during its transmission, falls under the area of data integrity protection.

Different approaches to this type of protection exist, such as *Message Authentication Codes* which add to the message a digital signature (see 2.3.1) given a key and a message. One of the most well established Message Authentication Code mechanism is Hash-based Message Authentication Code [KCB97]. Nonetheless, other strong mechanisms also exist, such as Cipher Block Chaining Message Authentication Code, with very similar performances [DHV03].

In a simple manner and as an example, a message authentication code mechanism using hashes can work as such:

Existing a key that is shared between two entities, a message's hash can be computed. Then, that hash is attached to the message that one entity wants to send. Using the same hashing algorithm and the same shared key, the receiver can compute the hash of the message he received. If it matches the hash that arrived, it can be assumed that the message was unaltered and therefore maintained its integrity.

How robust this process is depends on the robustness of the underlying algorithms. Common algorithms historically used for HMAC include MD5, SHA1 and SHA256. It is relevant to point out that some of the hashing algorithms may no longer be secure, since some have been demonstrated as vulnerable, such as MD5 and SHA1, demonstrated in [WY05].

2.3.4 SSL/TLS

[DR08] is the corresponding *Request For Comments* (RFC) for TLS 1.2, which is the most recent stable version of SSL/TLS. Version 1.3 is under development but it is not yet widely available, so the focus will remain on 1.2.

SSL/TLS is a mechanism that attempts to assure data integrity and privacy in the communication between two applications, using mechanisms such as the ones referred in sections 2.3.1, 2.3.2 and 2.3.3. This process is also able to, under some assumptions such as the ones in 2.3.2, authenticate at least one of the parties in the communication, usually the server in a client-server scenario. SSL/TLS also supports mutual authentication, where the client can also be authenticated, but this is seldom used.

On a mutual authentication scenario, a typical SSL/TLS handshake would follow the steps outlined in figure 2.1. Please note that some messages in a standard handshake were suppressed because they can be either optional, or too technical for a conceptual understanding.

The most relevant aspects of each message are:

- Client Hello: Specifying the highest SSL/TLS version that the client can support as well as some supported cipher algorithms.
- Server Hello: Chooses, from the above Client Hello, the algorithms and versions to be used.
- Server Certificate: Server sends its digital certificate to prove its identity (see 2.3.2)

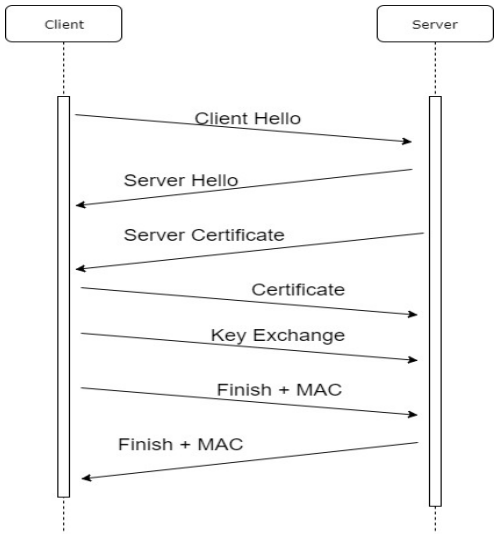


Figure 2.1: SSL/TLS Handshake with mutual authentication (simplified)

- Certificate: Client's Certificate (usually sent when explicitly requested, in order to handshake a mutual authentication session).

- Key Exchange: Define a key to be used in the communications. The way key exchanging is performed varies depending on the algorithm being used and may include previous server messages that were omitted. Consistently, this message will be encrypted with the public key of the server that is declared on the server's certificate. Thus, an approach as the one suggested by the end of section 2.3.1 is applied.

- Finish + Message Authentication Code (e.g. HMAC): Both client and server confirm the end of the handshake, and send a message authentication code of the messages sent before. The receiving end will use this message to confirm data integrity and as a confirmation of the successful exchanging of the keys and agreeing of the algorithms to be used (see 2.3.3).

2.3.4.1 SSL/TLS Vulnerabilities

In [ESM+15], different vulnerabilities of the SSL/TLS protocol that were discovered over the time are documented. Being such a core protocol in so many instances of over-the-wire communications, it is under heavy scrutiny and requirements to evolve and adapt are being consistently made.

Examples of attacks are the old cipher suites rollback, where a man-in-the-middle alters the Client Hello's supported cipher algorithms to some outdated and vulnerable ones, allowing for subsequent messages to be decrypted by the attacker; other more recent attacks, are the BEAST and CRIME attacks. All of them are presented in the mentioned document.

This constantly changing landscape serves as a good indicator that updated versions of the protocols must be generally enforced, especially in scenarios where the security of communications is of utmost importance, such as Payment Card Industry (PCI) or medical data scenarios. PCI's Security Standards Council even recommends in [Gra15] that by June 2018 servers should be fully supporting TLS version 1.1 or higher.

These recommendations and threats are the reasoning behind the focus on TLS version 1.2 or 1.1, the best suited versions of the protocol to be implemented in the project.

2.3.5 Testing tools and guidelines

Open Source Security Testing Methodology Manual (OSSTMM) [Her10] and Open Web Application Security Project (OWASP)'s testing guide v4 [MM14] present very detailed standards and guidelines to testing different security scenarios, including communications security in a process such as SSL/TLS. OSSTMM presents great theoretical guidelines and OWASP's testing guide goes as far as to suggest some different tools specifically for SSL/TLS testing like SSLScan[2]. These may prove very helpful for an efficient and thorough examination of the SSL implementation, which can often be a point of failure in an otherwise very strong protocol.

2.4 Data Mining

Data mining is defined by [HMS01] as a set of processes that are able to extract from usually great amounts of data different perspectives on its visualization that may prove useful. This includes simple statistical studies accompanied by enlightening data visualization techniques, or characteristics of the data such as interesting patterns and even inference of models that can be used for tasks such as predictive ones.

The context of the dissertation work lies heavily on data mining and this area will shape the security methods implemented by defining the shapes and types of information being transmitted as well as the threats that have to be addressed. Therefore, a solid understanding of the techniques that are used or can potentially be implemented in the applications is fundamental. In this section, different sorts of proposed techniques and overviews will thus be expressed.

2.4.1 Standard Process for Data Mining

[WH00] documents a CRoss Industry Standard Process for Data Mining (CRISP-DM) and is one of the standard guidelines for the development of data mining applications of many sorts.

As its core, CRISP-DM recommends some steps that should be followed for an efficient data mining development process:

- **Business Understanding:** Should be the first phase of the project and consists in understanding in what businesses or areas the project belongs to and in obtaining a solid grasp of

[2]available at https://code.google.com/archive/p/sslscan-win/

the objectives and hardships specific to the determined areas, often in the perspective of the end user.

- **Data Understanding:** This is a step where the data available is analyzed in a strictly observational way. Data is not altered, but important statistics are extracted, and relations between data, such as differences between different tables, are also identified. Identifying data quality problems that may exist, such as missing values, can also be useful in this step.

- **Data Preparation:** Here, the initial unaltered data suffers different operations such as handling of missing values, readjustments of the dataset structure and even addition of new data or metadata, in order to solve problems identified during the **Data Understanding** step and as preparation for the phases yet to come that may require some specific formats or characteristics of the data.

- **Modeling:** During modeling phase, different algorithms and techniques can be applied to try and extract models from the prepared data that can prove useful in tasks such as prediction.

- **Evaluation:** In this step, there must be methods to evaluate the performance of the process so far. For example, evaluating the quality by accuracy of the predictive model created during the previous phase and — in a contractual scenario — presenting the results to the product owner to evaluate the achievement of the defined objectives.

- **Deployment:** Deployment is always the last part of the process and should not be disregarded. Here, the results of the whole process have to be delivered to the product owner and integrated, for instance, on the product owner's systems.

The phases of CRISP-DM can and often will have some cyclical behaviors as, for example, new modeling techniques may demand additional data preparation or the results feedback during evaluation step by the product owner may be negative and mandate an improved business understanding. Figure 2.2 displays the different steps with common cyclical connections.

2.4.2 Data Mining Tasks

[HMS01] divides the data mining tasks in 5 different ones: density estimation and clustering, classification, regression, pattern discovery and retrieval by content.

Among them, two of them will be expanded here: clustering and classification.

2.4.2.1 Clustering

Clustering as defined by [JD88] consists in defining an abstract structure for a given dataset that is able to group different records of the data in different parts (clusters). This means that all records contained by a cluster are somehow similar to each other, while records from a different cluster are

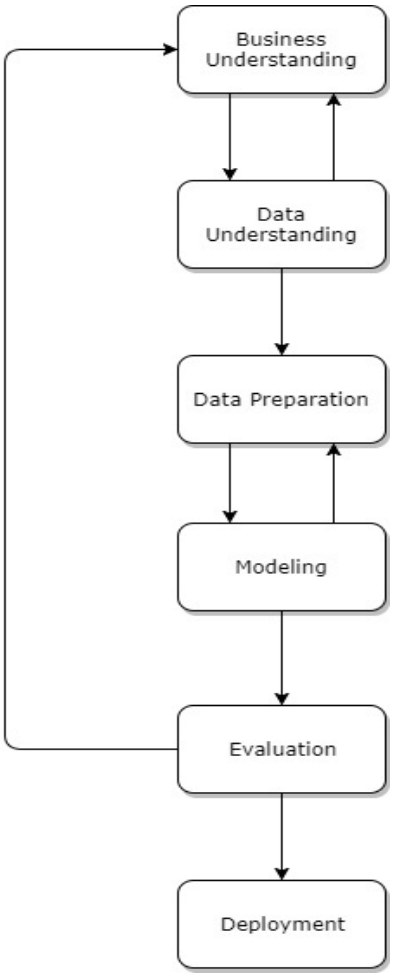


Figure 2.2: CRISP-DM cyclical behavior

not similar to them. Sometimes this abstract structure may not necessarily be on the same plane and may have a hierarchical shape.

Exemplifying, let there be a dataset of 12 individuals with information about their weight in kilograms and height in centimeters. Using a distance-based clustering algorithm such as k-means detailed in [KMN+02], and setting the value of k to 4, parameterizing the algorithm to search for 4 different clusters, the resulting clusters can be the rectangles represented in figure 2.3.

Interpreting the resulting clusters, 4 different groups of individuals are quite distinguishable in the dataset: less heavy and shorter individuals, less heavy and taller individuals, heavier and shorter individuals and, finally, heavier and taller individuals.

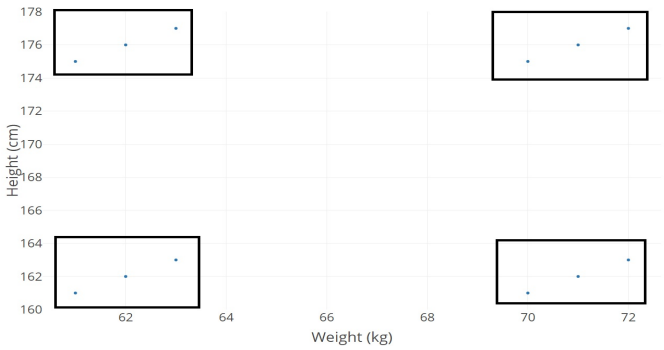


Figure 2.3: Different clusters (squares) of 12 records (dots)

2.4.2.2 Classification

Classification processes usually come along with predictive modeling. As opposed to clustering that tries to create descriptive models of the data that present useful or interesting information about the data, predictive models are models that allow forecasting of future events. The concept of classification comes when, by resorting to a predictive model as basis, a class is attributed to a new record. Which classes can be applied vary with each problem and may consist of a binary scenario where only two classes exist (e.g. in a context of bank transactions, a transaction being classified as one of two classes: *fraudulent* or *non fraudulent*) or may consist of a multi-class scenario (e.g. classifying the species of a flower, where the species are the classes).

The construction of the predictive models is made by analyzing a training set, i.e., a dataset of past records. These historical records may have an attribute that indicates which class they belong to (called label) in the case of supervised learning. When the label does not exist, the concept is named unsupervised learning[3].

An example of a supervised learning algorithm is decision tree learning, detailed in [Dob09], where the predictive model formed is a simple binary tree with the leaves being the classes. Figure 2.4 displays a decision tree as a predictive model used to attempt to predict the species of a flower. It was trained in a supervised setting with the iris dataset (a very commonly used dataset for demonstrations introduced in [Fis36]) containing information about a flower's sepal and petal width and length. The classes, i.e., the species that a flower can belong to, are *iris-setosa*, *iris-versicolor* and *iris-virginica*.

As it is observable in the decision tree, the above mentioned attributes of a flower can be used to attribute a species (or class) to it, simply by following the conditions set by the decision tree starting from the root node, until a leaf is reached.

[3]Clustering and outlier detection techniques are examples of unsupervised learning.

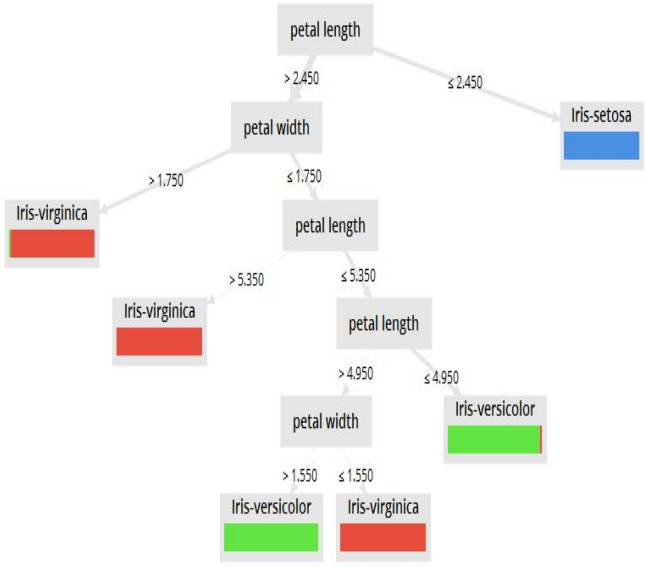


Figure 2.4: Decision tree for the iris dataset

In an identically supervised scenario, evaluation of the performance of the model is often done through splitting the original data into a *training set*, which is used to create the predictive model and a *testing set* (not included in the training set), solely for the purpose of evaluating its performance by inputting the attributes of each record into the model and comparing if the classes predicted by the model match the classes that were on the (omitted when input) labels of the training set.

2.5 Incremental Learning

Incremental learning (sometimes called online learning) introduces a new paradigm that tries to tackle the huge amounts of data available that may turn its simultaneous storage in memory at one given time unfeasible, as well as handling scenarios where data may not be simultaneously available.

Instead of a training dataset being readily available, and input into an algorithm so that, after training time, a model is created, incremental learning algorithms work on streams of data, learning from successive small batches of records or even individual records, and thus creating new models.

[DC03] proposes an adaptation of the standard *Support Vector Machine* (SVM) classification algorithm to support incremental learning and [Z+16] demonstrates that unsupervised learning methods can also support incremental learning, by proposing an incremental clustering method.

In [LHW18], a review and comparison between several state of the art incremental learning algorithms is made, where some metrics and properties are measured and documented, such as

accuracy, convergence speed, training time, ability for endless learning and viability for concept drifting[4].

Amongst the compared algorithms in [LHW18], incremental SVM was the best performer in terms of accuracy and convergence speed but displayed long training times, inability to endlessly learn, inability to adapt to a possible concept drifting and is inherently a complex model. Differently, *Stochastic Gradient Descent* displayed fast training times and the ability to endlessly learn and adapt to concept drifting, at the cost of its accuracy when compared to the other state of the art algorithms.

The same document provides a link to the implementations of the algorithms, including some provided by the package Scikit-learn for the programming language Python presented in [PVG+11].

2.6 Conclusions and Considerations

A careful evaluation of the dataset to be anonymized enables a correct selection of the methods to be used during its anonymization which minimizes the loss of data. High maintenance of the value of the data will boost the potential usefulness of data mining processes that thrive on big amounts of data, especially on incremental algorithms that are able to handle those amounts, through streams of data.

Finally, the secure transmission of data must serve as a very solid property of the developed platform. This is to be achieved by using highly established and proven methods such as TLS, while keeping its implemented version up to date, in order to tackle novel vulnerabilities.

[4]Concept drifting is the possible shifting over time of the relation between the attributes of a record and its target class that is attempted to be predicted

Chapter 3

Overview of Methodologies and Architecture

The solution consists of a prototype that serves as a proof of concept for the techniques implemented. This chapter will present an overview of the methodologies and explain the architectural setting, when relevant.

In short, the proposed solution revolves around dataset analysis and anonymization, implementation of secure conditions for data transmission between different machines and implementation of incremental machine learning techniques. Some processes are accompanied by appropriate testing and results evaluation.

3.1 Use cases and architecture

The actors of the solution are peers in a distributed setting. They can all act as client or servers, depending on if they intend to fetch some information like an anonymized dataset or data mining model (client), or if they intend to make either of them available to other authorized peers (server).

The first use case (illustrated in 3.1) addresses the scenario where a dataset is to be shared between the peers. In this situation, proper anonymization techniques must be applied to the dataset before it is made available. After the peer has anonymized the dataset, it must store the dataset in its server – usually divided in ideally-sized batches – and add to the server's configuration a list of the allowed entities that can fetch the information.

The second use case (illustrated in 3.2) behaves similarly but the shared information consists of a data mining model. All the sensitive information is used locally, by each peer's own local processes, thus it is always utilized by a peer-entity that is presumed to have permissions to hold and handle that information. Through this process, dataset privacy concerns are not relevant and incremental data mining techniques are at the core of this whole architectural setting.

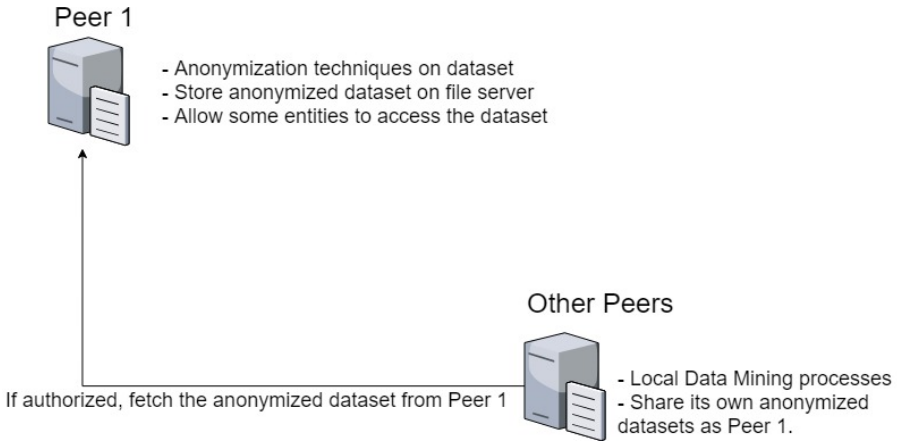


Figure 3.1: Use Case 1 - Anonymized dataset sharing

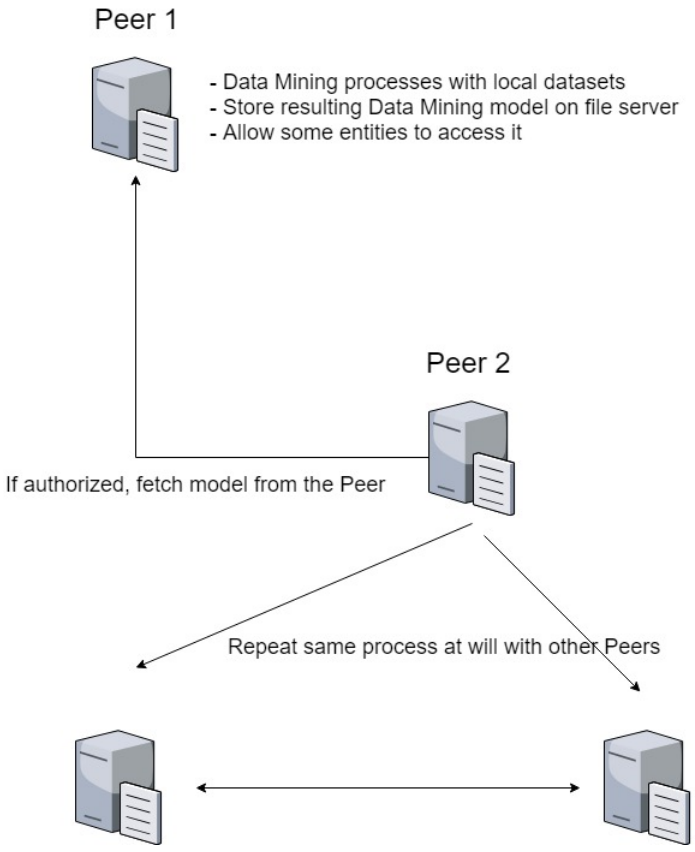


Figure 3.2: Use Case 2 - Data mining model sharing

Relevantly, all the data transmissions between the peers are performed under secure channels, as described throughout the secure data transmission categories of the dissertation.

3.2 Data analysis and preprocessing

Given that the usage of incremental data mining has the distinguishing feature of dealing with big amounts of data in a streaming format, the utilized datasets should be of a sizeable dimension, to fully take advantage of the algorithms being used.

After selecting the dataset to be utilized, it is analyzed and well documented to give sufficient knowledge as basis for future decisions, such as what anonymization steps should be taken or any further preprocessing.

There is a focus on descriptive statistics that can illustrate the generated dataset, which is a good way to summarize large amounts of data. Additionally, the identification of different types of attributes, including relevant direct or indirect identifiers for data anonymization purposes should be performed as well as appropriate correction of some values that may not be aligned with the desired format.

As a final note, implementations of data mining techniques could suggest some underlying additional revisiting of the data analysis and preprocessing. This aligns with the guidelines for the development of data mining applications suggested by CRISP-DM, presented in the bibliographic review.

3.3 Data Anonymization

A good analysis of the dataset proves to be pivotal in order to minimize the data loss in the process of anonymization.

As reviewed in the literature, the identification of the types of attributes that can serve as direct or indirect identifiers determines the techniques that can be applied to anonymize the dataset.

The goal of this portion of the work is to achieve an anonymized dataset through the usage of generalization, suppression and masking techniques that output a resulting dataset which meets acceptable risk thresholds.

After identification of the relevant attributes, direct identifiers have to be handled through masking or suppression techniques, with a preference for masking since it usually provides the least amount of data loss. Sporadically, some identifiers or records may prove too revealing, making them suitable candidates for suppression.

Handling indirect identifiers must be an iterative task, either manually or through computational iterations. Being generalization the main technique applied to indirect identifiers (that also provides the least amount of data loss), progressively stronger generalizations must be applied on each iteration while evaluating the resulting risk estimate.

For the risk metric, k-anonimity property reviewed in the literature will be used as a measurement of the risk of re-identification of the dataset. Acceptable risk thresholds for medical data can range between probabilities of re-identification of 0.05 and 0.33, as suggested by [EERM15].

Also during data anonymization, a more careful data analysis can be again performed. For instance, by studying the smallest equivalence classes in the dataset, it is possible to find that often

a single or a small set of records are inflating the risk estimation, and suppression of these records, or some of its attributes, may provide a much more data conserving anonymization than other alternatives such as going up on the hierarchy of a hierarchy-based generalization approach, while meeting the risk thresholds.

3.3.1 Tools

For most data handling purposes, the Python programming language is suitable for an efficient and flexible development.

Software applications such as RapidMiner (with documentation of its operators in [AH+12]) can be useful for some agile data visualization and analysis.

3.4 Data Transfer Security

The transmission of data will follow a client-server structure, where the client will be a machine that is requesting access to some given information, for instance the dataset that was anonymized earlier.

It is important that both client and server are authenticated, even if the client only provides some authentication after the initial handshake is made. Authentication of both parties is important so that the information origin is guaranteed, preventing false or corrupted information to be spread by malicious entities and authenticating the client guarantees that only authorized entities can receive the information.

For either the server authentication or both client and server authentications, digital certificates are created, with a fictional certificate authority as their provider. Since the scenario is setup as a proof of concept, the client and the server are to be configured so that this third entity is to be trusted as if it was truly a certificate authority. A scenario with mutual authentication in the TLS process is illustrated in Figure 3.3.

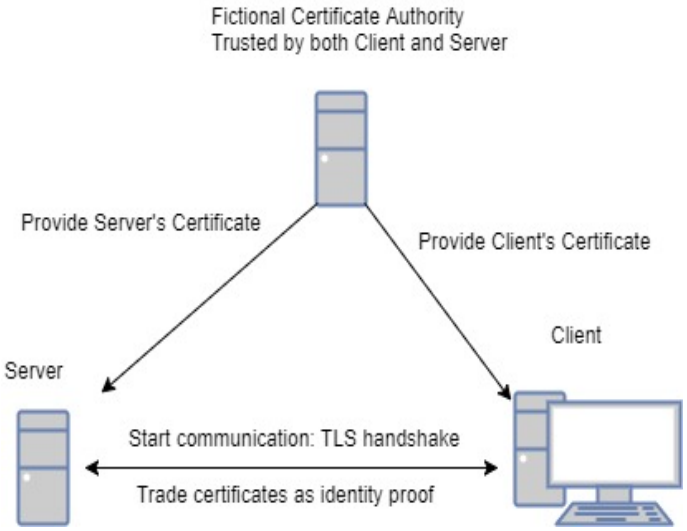


Figure 3.3: Trusted authority providing certificates and mutually authenticated TLS communications between client and server

As implied, TLS (versions 1.2 or 1.1, as justified in the bibliographic review chapter) will be the protocol used to guarantee the secure communications between client and server. As for implementation, the renowned library OpenSSL, further detailed in [VMC02], allows for efficient implementations of the SSL/TLS protocols and is under heavy scrutiny and well maintained, with relatively frequent updates. Although [AG07] portrays some other libraries as more efficient in terms of performance, OpenSSL shows portability and time efficiency in the implementation, while still being overall the most efficient in the group of the more flexible libraries. It is noteworthy that flexibility is an important trait when developing a proof of concept as is this dissertation work.

Certificates used in TLS processes contain information about the owner of the certificate, that can serve as basis for the ensuing authentication mechanism. Correspondingly, this sort of information will be the method to authorize other entities to access a server's model or anonymized dataset, as the client provides its certificate to present its identity and it is then matched against the allowed entities.

As for evaluation of the implementation, tools and guidelines presented in the literature review and other analysis tools should be used as they provide processes that can be very meticulous in the communications security testing, providing assurances of protection against the common attack vectors and assuring that the implementation of TLS protocol was executed under recommended and standard practices. Testing should be performed under a controlled environment as to minimize unexpected interferences and as to provide for an ideal attack scenario.

3.5 Data Mining

The final phase of the work consists in the implementation of different incremental machine learning techniques, focused on classification, as to provide a functional proof of concept, demonstrating that incremental learning processes can be integrated with the underlying implemented security and anonymization methods, maintaining meaningful results, when compared with standard processes.

Since the incremental algorithms handle as input a stream of data or small batches, the data is appropriately reformatted.

After the implementations of the algorithms (with assistance of different mentioned or reviewed tools and implementations such as *scikit-learn* package for Python language, RapidMiner and open-source implementations provided in [LHW18]), an empirical evaluation process has to occur.

This process of empirical evaluation revolves mainly around the comparison between the performance of the machine learning algorithms with the data stream consisting of the original data, stored locally in a single batch, and the performance of the same algorithms with data stream consisting of data after undergoing anonymization, possibly divided in small separated batches, and transmission processes.

A common evaluation method for classification algorithms has been described during bibliographic review. In short, a training dataset, where the class to which each record belongs to is known, is used as input for the predictions. Then, for each record, the model's prediction is matched against its known class, thus obtaining the total number of correct predictions, false negatives and false positives.

Chapter 4

Data Anonymization

Anonymization of datasets can be very useful in order to share information in a way that respects privacy between different entities. Motivated by the increasing awareness and legislation around data privacy – especially with medical data – this process gains relevance as a way to allow science to progress and continue its studies on these sorts of datasets, while respecting identities and other sorts of sensitive information. Furthermore, refinement and developments of anonymization processes and techniques may motivate or further reassure patients and other providers to allow their information to be used for research. It is noteworthy that, more than ever, considerable amounts of data can be used quite effectively, through the usage of computation and methods such as the ones used in Data Mining processes.

An important goal of data anonymization is to minimize the information loss of a dataset, while maintaining re-identification risks under an acceptable threshold. Because of it, some work has been done that phrases this goal as an optimization problem. Some algorithms have been developed, such as [SS98], that attempt to anonymize datasets while preserving the maximum amount of information. However, for this dissertation work, the process will be organic and iterative. This is to provide a more practical understanding of the decisions made and what steps were taken, that can empower further research with a better understanding of the techniques used by those optimization algorithms, and also to allow some more freedom on data loss selection, as it is not always true that minimizing the amount of data loss equals minimizing the loss of information value inside the context.

4.1 Dataset Selection

Searching for a usable medical dataset that was not yet anonymized – even if partially – proved to be a difficult task to accomplish on a public domain. Generating an artificial dataset was also considered but discarded since the anonymized dataset was to be later used for a predictive data mining task, which without real information can be devoid of value. Consequently, a more generic dataset was selected, that had demographic information about individuals, trying to resemble a record that, for example, an hospital would have about its patients. Much of the information

specific to the health domain such as cardiac frequency or blood pressure measures can usually be disregarded as identifiers, because most of this information is not apparent and can be quite hard for an attacker to obtain. Nevertheless, they must also be taken into account in medical datasets.

The chosen dataset was the popular Adult dataset, which gathers demographic information about over 32,000 individuals. This information is labeled, i.e., each record has a class attribute that associates it with either an annual salary equal or under 50,000, or over 50,000. Traditionally, this dataset is used in data mining exercises for a predictive classification task of exactly that: whether an individual earns more or less than 50,000 annually. Table 4.1 presents an overview of the several attributes of the dataset, including each attribute's type and example domain. Two fields have been omitted, since they are either redundant or meta information from data mining processes.

Attribute	**Type and Example Domain**
Age	Numerical: continuous
Workclass	Nominal: (Private, Federal-gov, etc.)
Education	Nominal: (Bachelors, Masters, 1st-4th, etc.)
Marital Status	Nominal: (Separated, Widowed, Never-married, etc.)
Occupation	Nominal: (Tech-Support, Armed-Forces, etc.)
Relationship	Nominal: (Wife, Own-child, Other-relative, etc.)
Race	Nominal: (White, Amer-Indian-Eskimo, etc.)
Sex	Nominal: (Female, Male)
Capital-gain	Numerical: continuous
Capital-loss	Numerical: continuous
hours-per-week	Numerical: continuous
Native country	Nominal: (United-States, Portugal, China, etc.)

Table 4.1: Adult dataset's attributes

All the records on the dataset were collected in [Koh96] and it was obtained from the UCI Machine Learning Repository, available at [DKT17], where the complete domain of the each attribute can be found.

4.2 Preparation

Reviewing, there are two different types of identifiers: direct and indirect (or quasi-) identifiers.

4.2.1 Direct Identifiers

Direct identifiers, by nature, directly identify the individual to whom a record belongs to, hence they should not be obtained in any publicly available dataset. Adult dataset is no exception, being in public domain, and does not contain any direct identifier (such as name, social security number, etc.). Nonetheless, it is relevant to mention how such types of attributes could be handled:

- **Supression:** Consists of completely removing the attributes that are direct identifiers. It is a simple approach as it does not include any other security concerns, but also implies the absolute loss of this information. Partial suppression can sometimes be applied — suppression of a portion of an e-mail address can sometimes suffice to remove that attribute off the category of direct identifier.

  Supression should be used when there is no interest in recovering the information or is crucial for security and privacy reasons that the information is never reaccessed or reobtained.

- **Masking or Pseudonymization:** At its core, masking features replacing the data with a random set of characters. It can come in several forms, for example through the usage of encryption, so that only an owner of a decryption key can access the original direct identifiers. Pseudonymization works similarly, by replacing direct identifiers with random codes. In the source, when pseudonymization occurs, pairs can be stored that record the original value of the pseudonym, allowing for recovery by contacting the original owner of the dataset, while also raising some security concerns regarding the storage and handling of these de-pseudonymization tuples.

  Masking or Pseudonymization should be used when it can be relevant to reaccess the original data of the direct identifiers. It may also raise some security concerns (whether in storage, key sharing and generation or true randomness of the characters generated) that should not be overlooked, to maintain the security and privacy of the whole process.

4.2.2 Indirect Identifiers

Since the Adult dataset is already void of direct identifiers, the anonymization process will revolve around handling indirect identifiers. Over these identifiers, two methods described in section 2.2.2 have been applied: **Generalization** and **Suppression**. Generally, generalization will account for the bulk of the changes on the dataset. More rarely, suppression is used for some problematic records, as removing a field of a small number of records may lead to less information loss than forcing all attributes of the dataset to broaden their meaning, incurring in a general loss of information throughout the whole dataset.

To finalize the preparation, it is necessary to identify the indirect identifiers and define a hierarchy for their generalization. The selected attributes and corresponding hierarchies are presented in table 4.2. Some of the chosen generalization hierarchy levels are based on analysis made to the dataset that is presented in section 4.3.

Although the field *Relationship* could be argued to be an indirect identifier, it contains either intelligible or redundant information, as combinations of other fields like *Marital Status*, *Occupation* and *Sex* already provide the same information (e.g. 'Wife' relationship is already indicated by Married' marital status and 'Female' sex).

Attribute	Level 1	Level 2	Level 3
Age	Round to 5	Decade	Round to 20
Education	ENUM(Basic, HighSchool, Superior)	-	-
Marital status	ENUM(Married, Divorced, Never-Married, Widowed)	ENUM(Married, Never-Married, Married-before)	-
Occupation	ENUM(Adm-manag, Technical labour, Care-oriented, Manual)	-	-
Race	ENUM(White, Amer-Indian-Eskimo, Other, Black)	-	-
Sex	-	-	-
Native Country	ENUM(America, Asia, Europe)	-	-

Table 4.2: Indirect identifiers and hierarchic generalization levels

4.2.3 Attack scenario and Risk thresholds

In section 2.2.3.1 two different re-identification attack scenarios have been described: Journalist and Prosecutor scenarios. Since the context of the dissertation involves healthcare related data and it is deemed as very sensitive, the Journalist scenario — where an attacker indiscriminately attempts to re-identify any record — will be the attack type that this anonymization process will attempt to prevent. This has implications in how the risk of re-identification of the dataset is calculated. Using the k-anonimity property and the concept of equivalence class where each class consists of a combination of indirect identifiers (see sections 2.2.3.2 and 2.2.3.3), in a journalist scenario, the risk of re-identification is given by equation 4.1, where $min(s)$ is the size of the smallest equivalence class in the whole dataset.

$$\text{Re-identificationRisk} = \frac{1}{min(s)} \tag{4.1}$$

Reiterating, section 3.3 mentions the acceptable re-identification risk for medical data as ranging between 0.05 and 0.33. This dissertation is selecting a middle ground of 0.20 as the acceptable risk threshold.

4.3 Dataset analysis

Statistical studies of the dataset were mainly performed with the help of Rapid Miner Studio[1], a tool for swift analysis of datasets and also used for data mining processes. The statistics in this chapter were, thus, produced with that software.

Analysis lays under the category of data understanding and is advised by CRISP-DM's methodologies (section 2.4.1) so this has been extrapolated to data anonymization context. In order for the analysis to be relevant, it should be centered mainly around the indirect identifiers, as they will be the subjects of generalization and suppression techniques.

[1] available at https://rapidminer.com/, last reviewed in 12 June 2018

4.3.1 Attribute analysis

4.3.1.1 Age attribute

Observable in figure 4.1, is an uneven distribution of ages between the individuals of the dataset. Although age ranges from 17 years old to 90 years old, ages over 67 account for a considerably small portion of the records. Since that range above 67 years old is so relatively unique, it may lead to equivalence classes that occur very rarely combined with other indirect identifiers, thus increasing the risk of re-identification of the dataset, as given by equation 4.1. Consequently, this age range may have to be handled, for example by grouping in a single value, to prevent it.

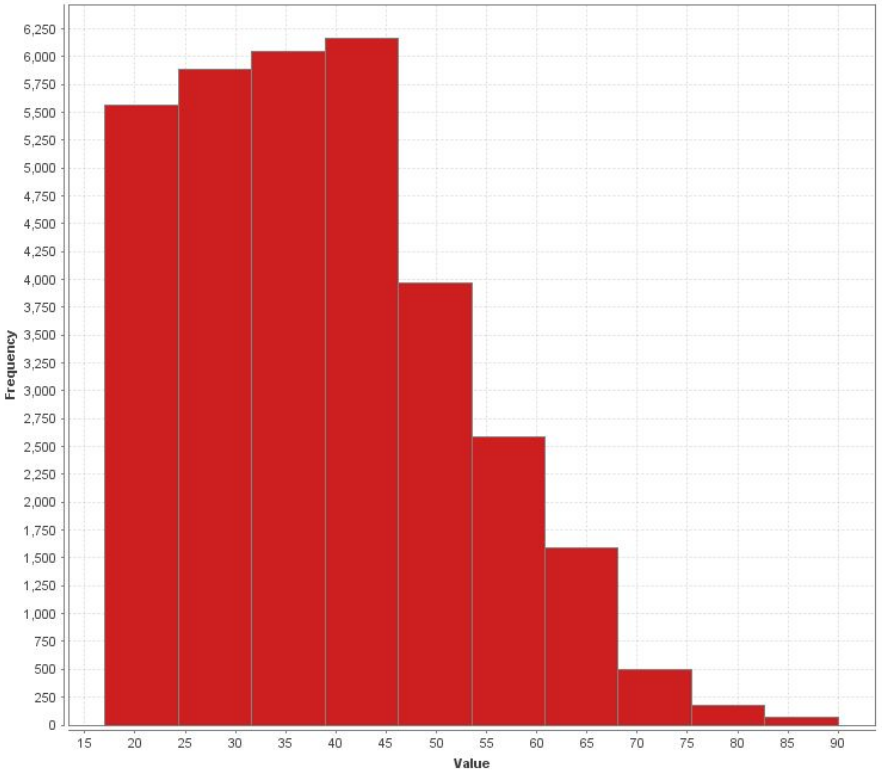


Figure 4.1: Age attribute values frequency

4.3.1.2 Education attribute

The frequency of several values of the Education attribute is considerably low (see figure 4.2). For similar reasons as mentioned in section 4.3.1.1, education column seems to be a good candidate for a generalization, given how it has a sparse distribution of values.

4.3.1.3 Marital status attribute

On the marital status attribute, some of its possible values presented in figure 4.3 have a low to very low rate of occurrence. This attribute is a suitable candidate for a generalization that can

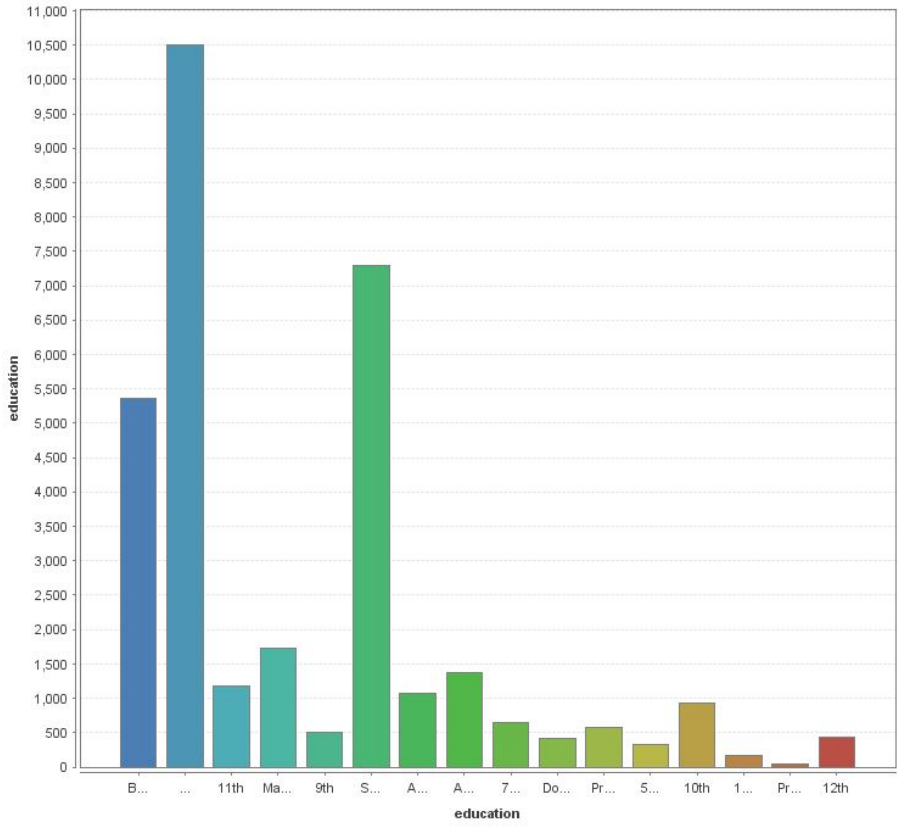


Figure 4.2: Education attribute values frequency

group these rare values into a more general one. As an example, Level 1 generalization hierarchy of table 4.2 of the *Marital Status* attribute replaces occurrences of the least occurring "Married-spouse-absent" (1,3% of the dataset) and "Married-AF-spouse"[2] (0,1% of the dataset) values and groups them into "Married", together with the most frequent "Married-civ-spouse"[3] (46% of the dataset).

4.3.1.4 Occupation attribute

Apart from a couple of values at the tail of figure 4.4, *Occupation* attribute seems fairly distributed. Suppression or a soft-generalization of the two problematic values may be due. Else, the attribute should only be affected if necessary by a big generalization step, because there is a considerable number of different possible values, even if evenly distributed in frequency.

4.3.1.5 Race attribute

Race can prove to be a difficult field to handle, given the huge predominance of 'White', making other *Race* values seem like outliers, with low frequency (support by figure 4.5). Also, at a seman-

[2] AF = Armed Forces
[3] civ = civilian

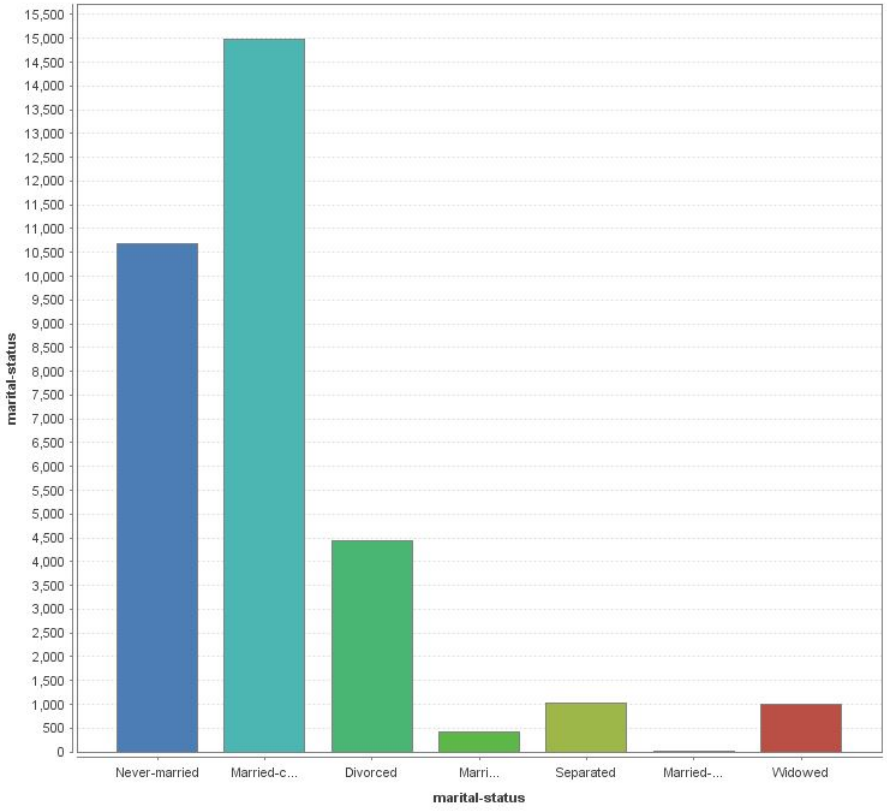


Figure 4.3: Marital Status attribute values frequency

tic level it is hard to label a superset (or generalization hierarchy level) that incorporates multiple *Race* values. A possible solution is to aggregate one of the outlier values into the "Other" nominal value.

4.3.1.6 Sex attribute

This attribute is very evenly distributed and both possible values (Male, Female) have considerably high frequencies (see figure 4.6). Hence, table 4.2 contains no predicted generalization hierarchy level as this attribute is not expected to suffer any generalization.

4.3.1.7 Native country attribute

Native country should prove to be another problematic attribute. There is a big presence of "United-States" values, but all other countries have a very low frequency, as Figure 4.7 shows that Mexico, the second most frequent country, has a relative frequency of 2%, compared to the overwhelming 89,6% of the United-States. This attribute should be the target of a considerably radical generalization, as suggested by table 4.2, where the first level reduces a big amount of possible values (countries) into a simple set of three possible values (continents America, Asia and Europe, where all the countries belong to).

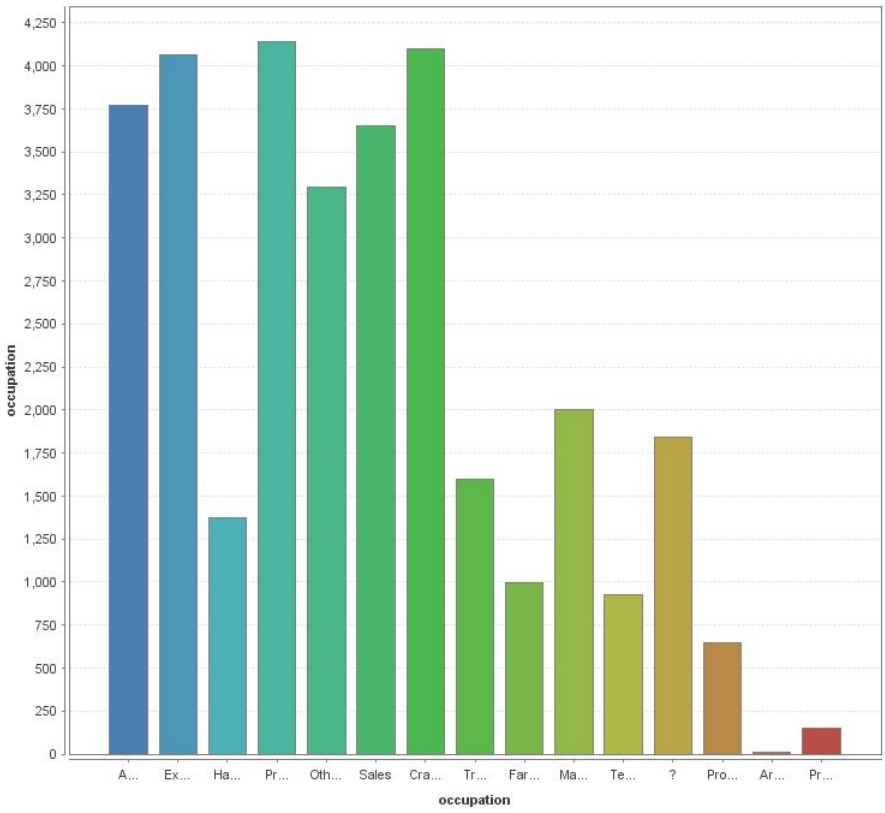


Figure 4.4: Occupation attribute values frequency

4.3.2 Re-identification risk calculation and problematic combinations identification

Section 4.2.3 defined the method to calculate the re-identification risk of a dataset in equation 4.1 and the acceptable risk threshold of 0.20 was selected (exact acceptable risk value is subjectively decided by each entity, as long as it is within the scientific consensus of the acceptable range of 0.05 to 0.30, for medical data). An interpretation of the equation and the risk threshold, is that the dataset meets the acceptable conditions when the following statement holds true:

"All possible combinations for the different values of each indirect identifier (each combination being an equivalence class) must occur at least 5 times in the dataset, when the combination occurs at least once."

So, the core goal of this entire process is to identify which equivalence classes occur less than 5 times and handle them either through generalization or suppression.

4.3.3 Algorithmic approach

Utilizing Python programming language and the well-known *pandas* library[4], the first challenge was to determine all the combinations of the indirect identifiers that occurred less than 5 times

[4] Available at `https://pandas.pydata.org/`, last reviewed in 12 June 2018

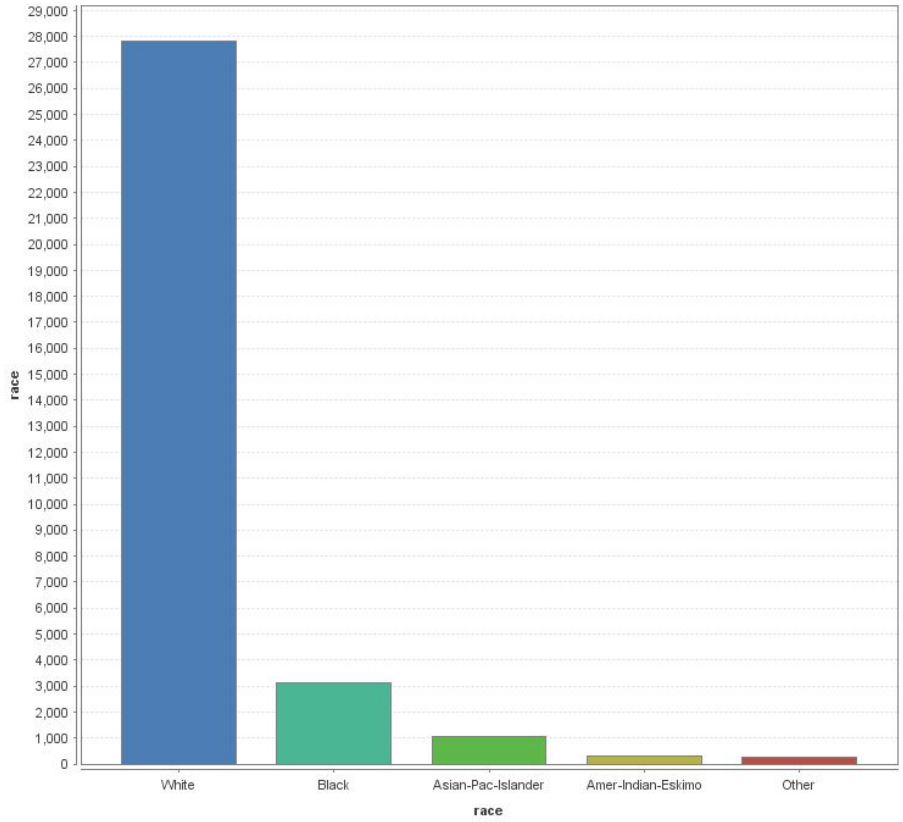


Figure 4.5: Race attribute values frequency

throughout the dataset.

For this, several approaches were tested:

1. **Iterating over all possible equivalence classes and counting their occurrences in the dataset:** This first tentative approach demonstrated that the process of going through all the combinations and over 32,561 rows of the dataset could be a time consuming process. Let the number of combinations from 2 to 7 elements of all the possible values for each indirect identifier be n and the size of the dataset be m: the time complexity of this approach is O(nxm). In this particular scenario, especially in the original unaltered Adult dataset, both n and m are quite large, resulting in a gigantic time complexity. Even when dividing the dataset into chunks or even if multiple threads were to be applied, this standard approach is too computationally demanding and, consequently, unfeasible.

2. **Standard iteration with tabu dictionary**: Iterating each element of the dataset, finding the differently sized (from 2 to 7) equivalence classes existing in each row and, finally, for each of the classes check the remaining elements of the dataset for the existence of this particular combination, until 5 occurrences are found or the dataset ends. To shorten this process, a combination is skipped if it already exists in the resulting "tabu" dictionary (meaning the occurrences of the combination have already been counted). This approach is still not ideal

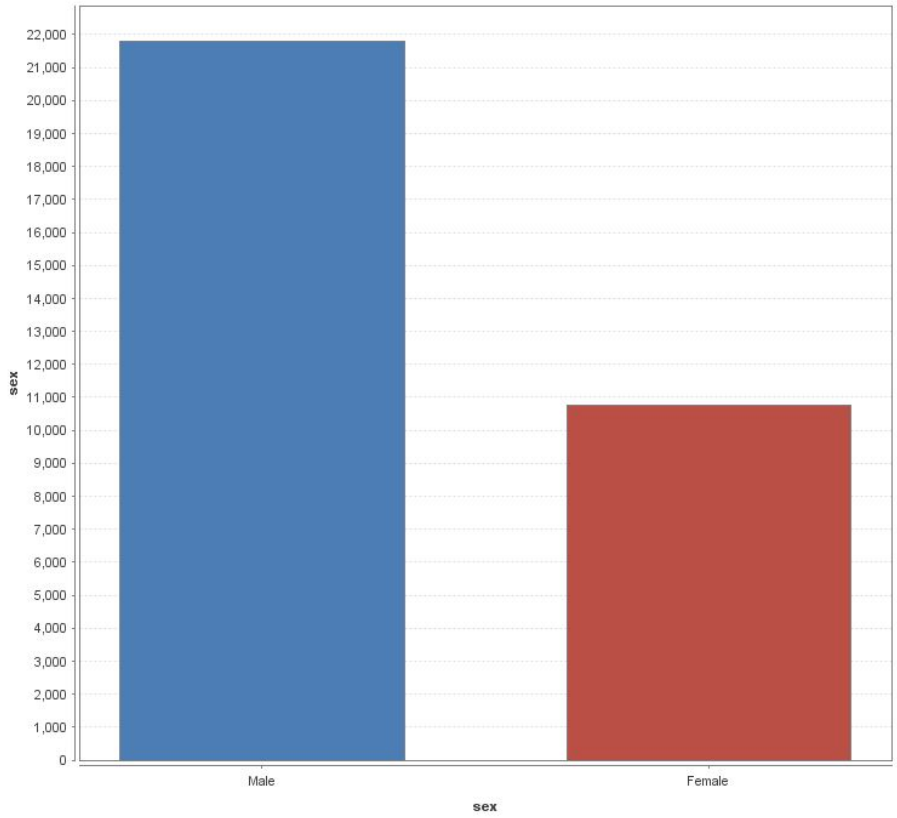


Figure 4.6: Sex attribute values frequency

Index	Nominal value	Absolute count	Fraction
1	United-States	29170	0.896
2	Mexico	643	0.020

Figure 4.7: Two most frequent native country attribute values

as shown by Annex A.1.1, where little over 1000 rows have been iterated over 10 minutes of runtime. The pseudo-code for this approach is represented in algorithm 1.

3. **Set intersection**: Sets are data structures that are very efficient by nature. For instance, both adding a new element and checking if an element already exists in a set have average time complexities of O(1). Since, so far, the bottleneck has been time complexity in the implementations, sets come as a tempting solution.

   Furthermore, at the core of this solution is the knowledge that an equivalence class with frequency equal or higher than 5 is cleared as non problematic.

   Assembling these two pieces, at its core, this method relies on firstly creating 5 different sets, stored in a list, followed by iterating over each row of the dataset and each equivalence class (from size 2 to 8) that each row has. Then, for each equivalence class, the class will

Algorithm 1 Iterate over dataset and combinations in each record, with tabu dictionary

```
1:  tabu = { }                                                    ▷ Dictionary
2:  for each index, row in dataset do
3:      for each integer i in range(2,7) do                       ▷ Combination size 2 to 7
4:          combinations = get-size-i-combs(row, i)
5:          for each comb in combinations do
6:              number-of-occurrences = 1
7:              if comb in tabu.keys then                         ▷ Combination already counted, proceed
8:                  continue
9:              end if
10:             for each remaining-row in dataset do              ▷ Rows not yet iterated over
11:                 if comb is-subset-of remaining-row then
12:                     number-of-occurrences += 1
13:                 end if
14:                 if number-of-occurrences >= 6 then
15:                     break
16:                 end if
17:             end for
18:             tabu[comb] = number-of-occurrences
19:         end for
20:     end for
21: end for
```

be added to the first set of the set list that does not yet contain that class. If all 5 sets contain that equivalence class, execution will proceed normally.

After the main iteration is over, some set operations will be performed, in order to return the classes that exist at least in one of the 5 sets, but not on all 5.

The pseudocode for this approach is presented in Algorithm 2 and its "EXTRACT-RESULT-FROM-SETS" procedure can help in providing a better understanding of the set operations done to extract the equivalence classes that do not occur at least 5 times in the dataset.

As seen in Appendix A.1.2, the execution time on the unaltered Adult dataset is a bit over 11 seconds, with the main iteration taking only around 9 seconds. This implementation is, then, viable and can help do relevant analysis of the dataset in a very acceptable time frame.

In summary, the third implementation presented was the only one that was viable, timewise.

Even after this result, some more pruning can be made. It is known that the superset of a set can never have higher frequency than the set. Or, in a different prism, let A and B be two sets and C be any other set; $A \cap B \cap C$ can never be greater in size than $A \cap B$. Contextualizing, if an equivalence class is a superset of another equivalence class with less elements, the bigger one can be pruned, as the superset is just an extension of the core problem that is the smaller equivalence class.

Algorithm 2 Approach using Sets and Set theory

```
    Output: result-set set with the combinations that occur less than 5 times on the dataset
 1: set-list = [set(), set(),set(), set(),set()]
 2: procedure MAIN-ITERATION:
 3:     for each index, row in dataset do
 4:         for each integer i in range(2,7) do                    ▷ Combination size 2 to 7
 5:             combinations = get-size-i-combs(row, i)
 6:             for each comb in combinations do
 7:                 for each c-set in set-list do
 8:                     if comb not in c-set then
 9:                         c-set.add(comb)
10:                         break
11:                     end if
12:                 end for
13:             end for
14:         end for
15:     end for
16: end procedure
17: procedure EXTRACT-RESULT-FROM-SETS
18:     set-union = set()
19:     for each c-set in set-list do                    ▷ get the union of all 5 sets
20:         set-union = set-union.union(c-set)
21:     end forset-intersection = set-union.copy()
22:     for each c-set in set-list do          ▷ Get set with elements that occur in all c-sets
23:         set-intersection = set-intersection.intersection(c-set)
24:     end for
25:     return set-union.difference(set-intersection)
26: end procedure
27: MAIN-ITERATION()
28: result-set = EXTRACT-RESULT-FROM-SETS()
```

The selected implementation does come with a limitation: its output is only a set with equivalence classes with frequency smaller than 5 and greater than 0. Even after pruning, more information is necessary to properly judge the dataset and to take an informed decision on what the next anonymization step should be. To do so, information described and exemplified in Appendix A.2.1 is collected by a Python script, allowing for better decision making on what process to use (generalization, suppression) and over which attributes or values.

4.4 Anonymization iterations

The process of applying generalizations and some suppressions is an iterative process. More relevant iterations will be described, along with the rationale behind each iteration's decision.

The first few iterations will only analyze and attempt to solve equivalence classes that occur less than 5 times with a size of 2. After some iterations, size of 3 will be handled and with due announcement all sizes will finally begin being handled.

Unless mentioned otherwise, generalizations applied will have been defined in table 4.2.

1. **Iteration 1**: The first analysis performed by the methods described in section 4.3 revealed a massive number of different equivalence classes that do not occur the minimum of 5 times, thus being below the acceptable re-identification risk.

   A big number of problematic equivalence classes warrants also a big generalization to be made. Since *Age* attribute has the biggest number of unique values (all integers between 17 and 90), it was selected to undergo the planned level 1 generalization.

2. **Iteration 2**: Annex A.2.2 displays the analysis result after iteration 1. It is clearly observable that the number of occurrences of problematic equivalence classes that contain a value from the *Native Country* is considerably higher than all other attributes. Furthermore, there is no country or small group of countries particularly more frequent in these problematic equivalence classes — except for the extremely common "United-States", all other countries occur fairly even (and low) amount of times.

   Also supported by *Native Country* being the attribute with more different values after iteration 1, it is justifiable to apply the specified level 1 generalization for this attribute.

3. **Iteration 3**: The number of occurrences of each attribute seems quite evenly distributed on Annex A.2.3, so level 2 generalization for *Age* attribute was selected, given that it was one of the softest generalizations that should amount for the least loss of information, mainly due to its numeric nature.

   Nevertheless, an immediate analysis after this generalization, documented on Annex A.2.4, displayed low effectiveness on this choice, so the generalization suffered an immediate rollback.

   It was then decided to go for the second attribute with most occurrences, which was the *Education* attribute. Further study revealed that there was no particular set of values that was especially problematic, so level 1 generalization for this attribute was applied.

4. **Iteration 4**: Although age level 2 generalization was reverted in iteration 3, Annex A.2.5 suggests that this generalization is indeed necessary, as *Age* is still a problematic field. Annex A.2.6 displays immediate results and further analysis reveals that, as predicted in section 4.3.1.1, ages over 60 were very unique and a big portion of what was disabling some equivalence classes to meet the 5 occurrences threshold. In consequence, ages over 60 were grouped into 60, with the meaning of "60+".

5. **Iteration 5**: Special attention on the counters of each value of Annex A.2.7 exposes a problematic value of 'Married-AF-spouse', in the *Marital Status* attribute. The value is an outlier among all other analyzed values, as predicted *a priori*, during analysis of the distribution of values of this attribute. The relatively soft Level 1 generalization for *Marital Status* was applied.

6. **Iteration 6**: This iteration consisted of some trimming, as few problematic equivalence classes of size 2 remained. 'Armed-Force' occupation only occurred 9 times throughout the whole dataset, so it was soft-generalized into 'protective-serv' occupation.

   Then, all the other problematic pairs of values were identified and manually handled by suppressing (swapping with missing data value '?') the most cells of the dataset that would minimize the number of cells affected, i.e., when a row has two problematic pairs, check if there is an element that exists in both pairs. If this holds true, replace that common element, thus solving two problematic pairs at once.

   Iteration 6 successfully solved all equivalence classes of size 2 and further iterations will account for pruned equivalence classes of size 3.

7. **Iteration 7**: With the sets of size 3 that do not meet the thresholds, *Occupation* attribute was revealed by the output in Annex A.2.8 as the the most occurring attribute in these sets. Level 1 generalization was applied. The different new values where each old value was grouped to is presented in table 4.3.

Group	**Value**	**Value**	**Value**	**Value**
Tech-labour	Tech-support	Craft-repair	Machine-op-inspct	Transport-moving
Care-oriented	Other-service	Sales	prof-specialty	Protective-serv
Manual	Farming-fishing	Priv-house	Handlers-cleaners	-
Adm-manag	Exec-managerial	Adm-clerical	-	-

Table 4.3: Generalization grouping of *Occupation* values

8. **Iteration 8**: Even though *Race* is a more frequently problematic attribute than *Marital Status* (see Annex A.2.9), values of *Race* are semantically hard to generalize, so, since the difference in frequency is negligible, *Marital Status* was chosen for a level 2 generalization.

9. **Iteration 9**: Given a small number of occurrences of size 3 after iteration 8, from this iteration onwards, all sizes of equivalence classes will now be handled. The total number of sets that still have to be handled is now 1293. Out of those, 1024 contain a value from the *Age* attribute. Even if incurring in considerable information loss, it is necessary to protect the privacy of the dataset to generalize the attribute into level 2 of the generalization hierarchy.

10. **Iteration 10**: The output in Annex A.2.10 reveals a problematic value of 'Asian-Pac-Islander' in the attribute *Race*. Level 1 generalization where this value is grouped with the 'Other' value is applied.

11. **Iterations 11-15**: These final iterations will revolve around substituting key problematic cells with missing value sign '?'.

    Firstly, all the problematic sets are stored and will be iterated over. If a problematic set exists in a row of the dataset, a dictionary with the index of the row as key will store the

set's elements as value. If this row already had one or more elements as value, the new values will be appended to the list (even if repeating).

Secondly, for each entry of the dictionary, the most frequent element of each row of the dataset will be defined, along with all elements that occur the same amount of times.

Finally, one of the determined more frequent problematic elements on a row will be replaced, depending on which attribute they belong to, abiding by the following priority:

Native Country > Race > Marital Status > Education > Occupation > Age > Sex

Being *Native Country* the first option to be replaced and *Sex* the least preferable one [5].

After some iterations (since each iteration conservatively only suppresses the most frequent problematic elements), the anonymization process has been concluded and all equivalence classes now feature a frequency of at least 5 on the dataset.

4.5 Process evaluation

[[CKLM09], chapter 3, *Utility Metrics*] proposed a *Loss Metric* (LM) to evaluate the information lost in a generalization. LM works as follows:

Let A be an attribute, tA be a tuple of values of A and x be the generalization value to which members of tA were transformed to.

If $|A|$ is the size of the domain of A, i.e., the number of all possible pre-generalization values, and M the number of values in tA, then the LM for a generalization tA -> x is given by equation 4.2.

$$LM_x = \frac{(M-1)}{(|A|-1)} \tag{4.2}$$

In [CKLM09], LM for each attribute A, LM_A, is given by the average of all LM_x for all generalizations x performed for each tuple tA.

Although this average presents a very interesting metric that can easily be applied to generalization processes in data anonymization, it only accounts for the number of different values that were grouped into new generalization values, with equivalent relevance. Meanwhile, it does not take into account the amounts of data that were altered, i.e., how many rows of the dataset were affected by a generalization x. So, to proportionally change the relevance of information losses of generalizations that affect more or less rows, the weighted mean is used to determine LM_A.

[5] Since the context of the dissertation is inserted in a medical scenario, this priority order was defined by attempting to predict which information would be the most relevant for medical purposes, but the relative value of each attribute is not absolute and is subject to different interpretations depending on the exact final purpose of the information.

Let x_i be a generalization for tuples belonging to attribute A, let R be the number of rows in the dataset and R_{x_i} be the number of rows affected by a generalization x_i, equation 4.3 determines the weighted LM_A, wLM_A.

$$wLM_A = \sum_{i=1} LM_{xi} * \frac{R_{xi}}{R} \tag{4.3}$$

After all the iterations described in section 4.4, the weighted loss metric for each attribute has been calculated and is represented in table 4.4.

Attribute	**Loss Metric**
Age	0.318
Education	0.254
Marital Status	0.132
Occupation	0.204
Race	0.013
Sex	0.0 (no generalizations)
Native Country	0.566

Table 4.4: Loss Metrics for the different indirect identifier attributes.

As expected, *Race* attribute did not incur in a big information loss, using weighted Loss Metric as a measurement, because only one possible value with a low number of occurrences throughout the dataset was generalized.

Native country seems to have taken a considerable amount of information loss and it can be justified by the great disparity of sizes of the domains of the attribute before and after its generalizations. Coincidently, it was also the first attribute to be removed on the default priority of the applied suppressions.

Overall, the Loss Metric values achieved are acceptable, while applying some intuitive generalizations to the data, maintaining its ease of understanding.

4.6 Considerations

Organically optimizing the anonymization of a dataset requires a good amount of experience, understanding of the data and of the context, and solid processes to obtain the information to support the best decisions. When reapplying this process, a version control environment is advised, to swiftly allow for some possible rollbacks to previous states.

Any future attempts to automate the data anonymization process require solid understanding of the techniques and decisions made during an organic process, as well as being accompanied by careful and gradual evaluation of the state of the dataset because, as seen during the organic process, sometimes the best decisions or the next steps may have not been predicted *a priori*, as was the case with some soft-generalizations or occasional suppressions that were not established during the preparation phase.

Chapter 5

Data Transmission Security

For the purpose of sharing information either of an anonymized dataset or of a data mining model to be used in a distributed setting, most if not all of the peers in the system must be able to behave as a server to propagate their information and simultaneously as a client, to be able to obtain information from the other peer/sources. These channels of transmission must be secure, as medical information is quite sensitive and in many situations must not be made publicly available. Accessibility to the data is also important and adding it to the concern of only some selected entities being able to access the information that these channels can transmit, a secure authentication mechanism is also a necessity. When possible, it must also swiftly allow for new entities to request or access the data.

To ensure state-of-the-art mechanisms, section 2.3.4 justifies the usage of at least version 1.1 of the TLS protocol, to match current high security standards, together with a stable version of OpenSSL (oldest version tested in the dissertation work was 1.0.1t) for the certification infrastructure to be developed.

Additionally, being TLS a protocol under such heavy scrutiny, it can be considered quite sturdy, thus vulnerable contexts where faults can occur lie on both configuration and implementation of the protocol, that must be duly inspected for a good evaluation of mechanisms applied.

5.1 Certification Infrastructure

The usage of digital certificates usually relies on an agreement that one or a set of *Certificate Authorities* (CA) are trusted between all the users involved and can indirectly attest the identity of another entity. Since this dissertation work involves a prototypical implementation of different mechanisms, a CA has been created (named CAmed) and all machines will be configured in order to accept CAmed as a trustworthy authority. In a real world scenario, this CA would be a specialized company that is trusted world wide. A list of common CAs can be found in most browsers security configurations, as is exemplified in figure 5.1, displaying a portion of a list of CAs of Mozilla's browser Firefox (version 58.0 64-bit).

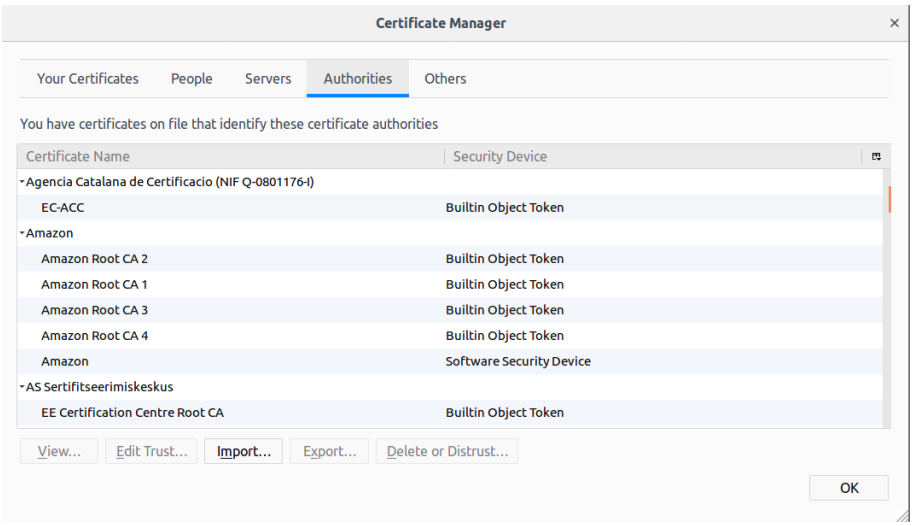


Figure 5.1: Mozilla Firefox Certificate Authorities list

After CAmed's creation, entities that want to obtain a digital certificate (both as client and server, since TLS with mutual authentication was applied) signed by CAmed must create a *Certificate Signing Request* (CSR) and send it to CAmed. Usually, this is where a CA verifies the claimed identity of the entity by verifying and confirming CSR's fields (such as *Organization*, *Common Name* or *Country*). Finally, if everything is validated, the CA sends a signed certificate generated from the CSR to the requester. Further details about the cryptographic mechanisms used to validate certificates were reviewed in chapter 2, more specifically in section 2.3.2, addressing *Digital Certificates*.

5.1.1 Creating a Certificate Authority

All commands and instructions described have been performed in a machine running Ubuntu 16.04 operating system.

1. **Machine preparation**

   The first step on creating a CA is proper configuration, starting by verifying or setting the hostname and the *Fully Qualified Domain Name* (FQDN) of the machine. Usually this is done by a DNS entry, or locally by editing an entry on the 'hosts' file usually located in /etc/hosts with the local permanent IP address (127.0.1.1), followed by FQDN and simple hostname (see Appendix B.1.1 for the file used during the implementation). Hostname and FQDN configuration should be followed by adjusting the time/date of the machine. The usage of Network Time Protocol (NTP) is recommended to automate this process.

2. **OpenSSL configuration**

   Some OpenSSL configuration is now required, starting by indicating the root directory of where CA's files will be stored. **It is recommended that this is stored under /root/ folder together with the following command:**

```
> chmod -R 600 /root/<ca_directory>
```

   This makes it so that superuser privileges are necessary to even access the directory of where sensitive files such as keys will be stored. Relevant sections of OpenSSL's configuration file that should be edited are described in Annex B.1.2.

3. **Root directory creation and composing**

   With sudo access, the root folder specified in OpenSSL's configuration file should be created, along with some sub-folders: newcerts, certs, crl, private, requests. These sub-folders will store newly generated certificates, any certificates found fit, hold certificate revocation list information, store private files such as private keys and store CSRs, respectively.

   Furthermore, empty 'index.txt' and 'serial' files should be created in the base directory. The first file will store information about generated certificates, while the second, initialized with an integer, will assign a serial number to each certificate and automatically update it to the next serial number, every time a new certificate is created.

4. **Private key generation**

   A private key, within the context of *Public Key Infrastructure* (PKI), is under the assumption that only the entity that has generated it can access it. **Loss or embezzlement of this key can lead to the inability to continue operations without a reset of the whole process or lead to another entity faking its identity as if it was the owner of the private key.**

   Given these concerns, OpenSSL can generate a private key, appropriate to the RSA algorithm. Since this key and its security is pivotal, a considerably great key size is recommended, of at least 4096 bits. Another secure step that should be added is the generation of a password, to be used with AES algorithm in the protection of access to the private key file. Commands for generating the key with these parameters are (on CA's root directory):

```
> openssl genrsa -aes256 -out private/cakey.pem 4096
```

   The command will then generate a 4096 bit long private key stored in 'private/cakey.pem'. Access to the file is password protected, so a prompt should appear for password creation, using AES256 algorithm.

Generation a public key, pairing with this private key, can be easily done at any moment, provided access to the private key is granted.

5. **Root certificate creation**

   A CA must itself possess a digital certificate, named a *Root Certificate*, usually self-signed, to allow the distribution of its public key.

   A possible command to create this *Root Certificate*, storing it in 'cacert.pem' file, with 10 years of validity can be:

```
> openssl req -new -x509 -key private/cakey.pem \
-out certs/cacert.pem -days 3650 -set_serial 0
```

   X509 is a standard for certificate file formats and the key used for (self-)signing the certificate is the private key generated above, that should prompt a password insertion request. Upon using this command, some information about the entity (CAmed) will have to be filled. Fields added to that prompt during dissertation work are specified in Annex B.2.1.

5.1.2 Generating Digital Certificates

In order for a peer or any entity to obtain a digital certificate, they must produce a *Certificate Signing Request* (CSR). In short, the entity claims who the entity is in the CSR and sends it to the CA for validation.

In traditional SSL/TLS mechanisms, the server is frequently the entity who must own a certificate to prove its identity. However, this scenario uses mutual authentication, where the client must also provide a valid digital certificate for client-server communications to go through.

First, any entity must possess its own private key. Refer to 'Private key generation' in section 5.1.1 above on how to do so and for some security concerns [1]. While naming the file where the private key is held 'peer_private_key.pem', a CSR can be created with the terminal command:

```
> openssl req -new -key peer_private_key.pem -out peer_req.csr
```

Some situations, like some use cases in the Google Chrome browser, may require the *SubjectAltName* field to be specified as well. Creation of a CSR with this field is exemplified and further detailed in Annex B.2.2.

Access to 'peer_private_key.pem' must once again require the password that protects the file.

Created CSR file should now be sent over to the CA that will produce a corresponding signed digital certificate. To do so, the CA must input the following command:

[1] Since in this case the private key is of a standard server or client and not as sensible as a CA's private key, it is currently tolerable for the key to have a smaller size, such as 2048 bits, instead.

```
> openssl ca -in peer_req.csr -out peer_cert.pem
```

Standard output of this command features is demonstrated in Annex B.2.3 and features a prompt to insert CA's private key password to access the file, display of the information of the entity that generated the CSR and a confirmation generating a new digital certificate and signing it. Again, this is where a process should be initiated by the CA to verify the information stated on the CSR by the requesting entity.

If the process went smoothly, after transmission of the generated certificate, the requester now holds a digital certificate that can be used to transmit its public key to other users that trust the same CA, and perform an authentication protocol.

5.1.2.1 PKCS12: Importing certificate into a browser

PKCS12 is an archive file format to store several cryptographic files into a single bundle. A PKCS12 archive is usually required to import a client's digital certificate into a browser. Specifically, both client's private key and client's digital certificate are bundled. Annex B.2.4 describes the commands and prompts to generate one of these bundles.

Most browsers should provide a graphical interface (usually in security / certificates settings) that makes importing this bundle into the browser a trivial task.

5.2 Server side

The server was built with the assistance of Apache 2 [2]. Scripts for operations regarding file manipulation were developed, including some configuration files editing and distribution of data between different directories of the file server.

A premise of this process is that the server already possesses a digital certificate, in this scenario validated and signed by the local CA, CAmed, that is trusted by all the peers.

5.2.1 Setup

Apache2 is required for the development of the server, so the first action should consist in using a package manager (either graphic of command line based) to install apache2.

Enabling and properly setting up SSL/TLS in apache2 requires some additional commands and steps, described below.

1. Enabling Apache's SSL module.

```
> sudo a2enmod ssl
```

[2] Documentation available at https://httpd.apache.org/docs/2.4/

2. Creating a symbolic link for the default-ssl configuration file.

```
> sudo ln -s /etc/apache2/sites-available/default-ssl.conf \
/etc/apache2/sites-enabled/000-default-ssl.conf
```

3. Adding key and certificate to default-ssl configuration file.

```
> sudo nano /etc/apache2/sites-enabled/000-default-ssl.conf

*Edit the following values:
SSLCertificateFile <absolute path to certificate>
SSLCertificateKeyFile <absolute path to private key>
```

4. Restarting apache server

```
> sudo service apache2 restart
```

   Since the private key is now used by Apache, the password to access its file should be requested upon restart.

Apache server's default 'it works' landing page should now be accessible through a standard browser both when using standard HTTP and when using HTTP with SSL/TLS (HTTPS), under the assumption that the machine that accesses the client trusts the CA that signed the certificate.

5.2.2 File structure

The way files are distributed in the server depends on what is the use case that was suggested in section 3.1, i.e., depends on if an anonymized dataset or a data mining model is being stored in the server.

Starting from the root directory of the server, data mining models are stored under it in the models/ folder. Meanwhile, anonymized datasets are under datasets/ folder, where a folder with the name of dataset is created and the dataset is divided in batches, each under a sequentially numbered folder, with the file name 'batch<sequence_number>.csv'.

An exemplifying file tree is displayed in figure 5.2.

While storing and making a data mining model available is a straightforward task, placing an anonymized dataset correctly divided in batches raises some concerns. The developed Python script can take the approximate size (in kB) of a batch and divides and stores the information in such a way that a batch does not end in between a row of the dataset.

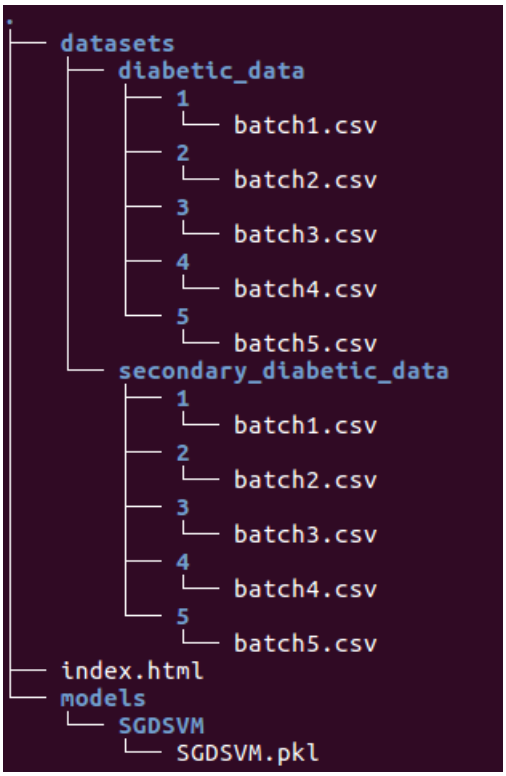


Figure 5.2: Server's file tree.

5.2.3 Security mechanisms

5.2.3.1 Enabling mutual authentication

In a similar way a server can use its digital certificate to prove its identity to the client, the client can also send its own digital certificate to prove its identity to the server (again, with the core assumption that both trust the CAs that signed the digital certificates). With TLS, this process is called mutual authentication.

Enabling mutual authentication in apache2 requires editing fields of the default-ssl configuration file of the server. The necessary editions are listed below:

1. Uncommenting SSLCACertificateFile and adding the absolute path to the trusted CA's certificate (in the developed scenario, CAmed's digital certificate file).

2. Uncommenting SSLVerifyClient require.

3. Uncommenting SSLVerifyDepth and editing its value to '2'.

5.2.3.2 Authentication forcing and managing

As client authentication is a required step to allow access to the data in the server, HTTPS access to the server has to be forced. HTTPS forcing is achieved by adding or editing a .htaccess file in

the server's root directory to match the contents in Annex B.1.3 together with adding the following to the virtual host in the default (non-secure) site configuration file (usually in Apache folder/sites-available/000-default.conf):

```
<Directory /var/www/html>
AllowOverride All
</Directory>
```

"/var/www/html" should be replaced by the server's root directory. Finally, enabling of Apache's rewrite mod and restarting of the Apache server may be necessary. These are the two commands necessary to do so:

```
> sudo a2enmod rewrite
```

```
> sudo service apache2 restart
```

Now that the client is forced to share its own digital certificate, the server can match its information to a list of entities that can access server data — after verification of the certificate's validity through standard initial TLS communications. Information that can be matched is any of the fields about the subject of a X509 certificate:

- Country
- State
- Location
- Organization
- Organizational Unit
- Common Name
- Email

For a quick addition of new allowed entities, the default-ssl configuration file has been altered so that a Python script can easily alter the contents of a separate file. This includes:

1. Removing SSLVerifyClient and SSLVerifyDepth lines
2. Replacing it with

```
<Directory "/var/www/html">
                        SSLVerifyClient require
                        SSLVerifyDepth  2

                        SSLRequireSSL
                        Include valid_expressions.list
</Directory>
```

where "/var/www/html" is the root directory of the Apache server and 'valid_expressions.list' is the file being altered by the Python script, on the root folder of apache (usually /etc/apache2).

With this configuration, the script will receive information about different fields of the entity to be added and edit 'valid_expressions.list'. Help about the usage of the script and an example file that matches some fields of a client certificate (valid_expressions.list) can be found in Annexes A.3.1 and B.1.4, respectively.

5.3 Implementation Evaluation

5.3.1 Experimental Setup

Three different machines were used to verify the implementation of SSL/TLS: a client machine (*tuxCLI*) and two server machines, porto.fe.up.pt and douro.fe.up.pt (*tuxSV1* and *tuxSV2*, respectively). *tuxSV1* and *tuxSV2* had their servers setup as described above, throughout section 5.2. *tuxCLI* also has a digital certificate of its own that can be used for client authentication.

Each scenario is matched against its expected behavioral outcome. Scenarios, parameters and predicted outcomes are documented in Table 5.1

Scenario	Client Certificate	Server Certificate	Expected Outcome
1	✓	-	Handshake terminated by client.
2	-	✓	Handshake terminated by server.
3	✓	✓	Successful access.
4	✓ but not authorized	✓	Rejection by server. (403 unauthorized)

Table 5.1: Experimental scenarios, valid certificates and expected outcomes.

Note: On scenario 4 the client has a valid certificate, but client's identity does not match any authorized identity on the server's allowed entities list.

5.3.2 Experiments Results

Results displayed have been gathered with the assistance of Firefox browser version 60.0.2 and network packet information through Wireshark [3] — adding to the previous technologies already mentioned such as OpenSSL and Python language.

Some preliminary steps have been taken before undergoing the different experiment scenarios, such as verifying Apache's server status, correct placement of digital certificates, keys and PKCS12 bundles and versions of the frameworks used. Most relevant preliminary information gathered was confirmation that the server supports the currently most recent TLS version (v1.2), with the commands and outputs on Annex B.3.1.

1. **Experimental scenario 1**:

   Even with *tuxCLI* having a proper digital certificate, communications with a server that possesses a non valid certificate should be closed by the client. For example, having *tuxCLI* not trusting the CA that signed the certificate of *tuxSV2* (CAmed), the browser displayed an error message and blocked access to the server (Figure 5.3).

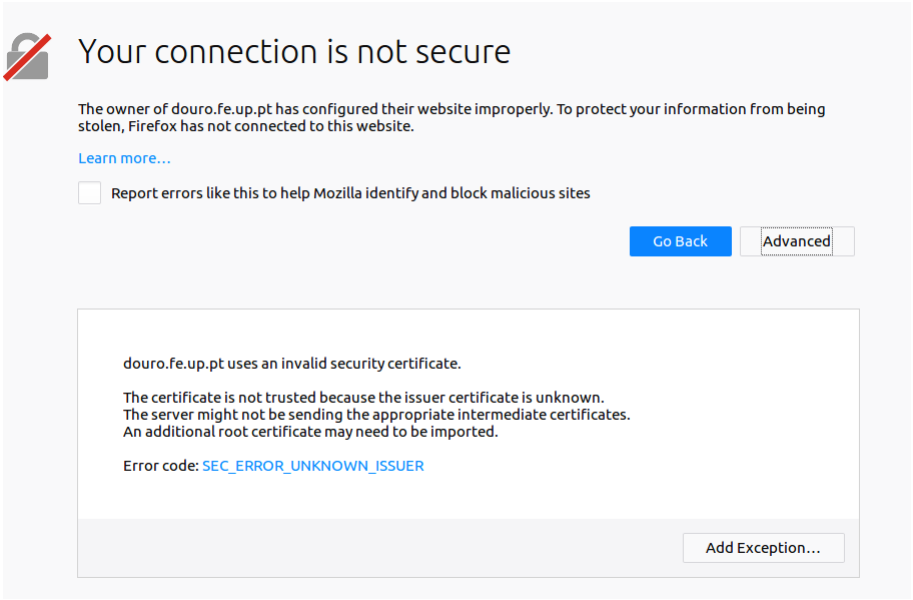


Figure 5.3: Client browser: invalid *tuxSV2* certificate

   Further network packet analysis can be found in Annex B.3.2.

2. **Experimental scenario 2**:

   In this scenario, the browser should validate communications and a normal use case of TLS would allow access to the server, but since mutual authentication is being used, the server will reject client access when either no client certificate is provided or the CA that signed it is not trusted by the server.

[3] Available at https://www.wireshark.org/

Figure 5.4 displays an error, 'SSL_ERROR_HANDSHAKE_FAILURE_ALERT', meaning that the initial TLS handshake was not successful. Further investigation of the network packets in figure 5.5 confirm that the server *tuxSV2* sent the first 'Encrypted Alert' message that prompts termination of the TLS handshake.

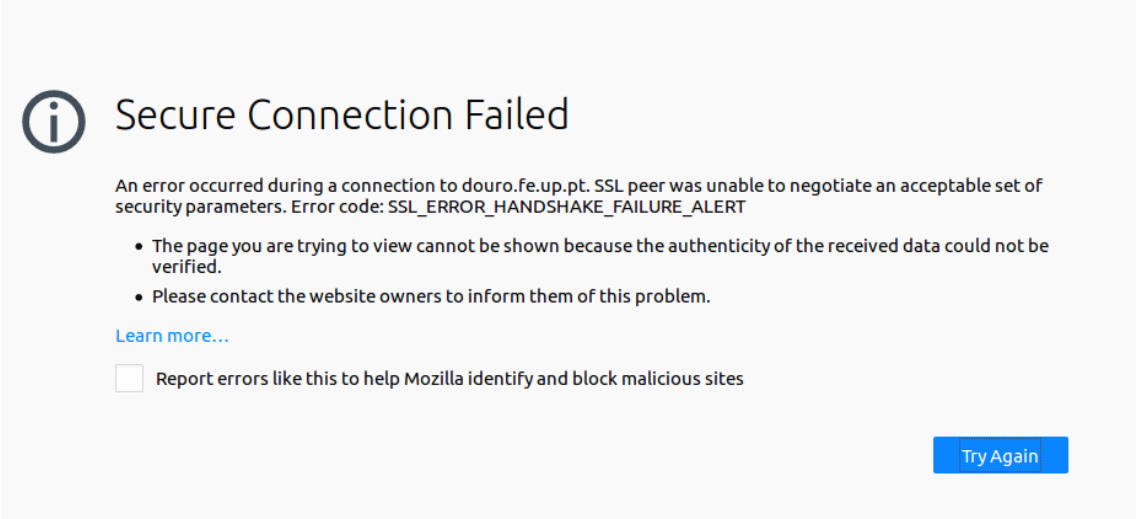


Figure 5.4: Client browser: invalid *tuxCLI* certificate

No.	Time	Source	Destination	Protocol	Length	Info
35	2.016599965	10.0.2.15	10.227.107.1…	TCP	74	59652 → 443 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3735208 TSecr=0 WS=128
36	2.017450177	10.227.107.1…	10.0.2.15	TCP	60	443 → 59652 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
37	2.017468752	10.0.2.15	10.227.107.1…	TCP	54	59652 → 443 [ACK] Seq=1 Ack=1 Win=29200 Len=0
38	2.018394628	10.0.2.15	10.227.107.1…	TLSv1.2	571	Client Hello
39	2.018652461	10.227.107.1…	10.0.2.15	TCP	60	443 → 59652 [ACK] Seq=1 Ack=518 Win=65535 Len=0
40	2.022578383	10.227.107.1…	10.0.2.15	TLSv1.2	2894	Server Hello
41	2.022590050	10.0.2.15	10.227.107.1…	TCP	54	59652 → 443 [ACK] Seq=518 Ack=2841 Win=34080 Len=0
42	2.022697823	10.227.107.1…	10.0.2.15	TLSv1.2	381	Certificate, Server Key Exchange, Server Hello Done
43	2.022703623	10.0.2.15	10.227.107.1…	TCP	54	59652 → 443 [ACK] Seq=518 Ack=3168 Win=36920 Len=0
44	2.023218513	10.0.2.15	10.227.107.1…	TLSv1.2	147	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
45	2.023398373	10.227.107.1…	10.0.2.15	TCP	60	443 → 59652 [ACK] Seq=3168 Ack=611 Win=65535 Len=0
46	2.024455723	10.227.107.1…	10.0.2.15	TLSv1.2	328	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
47	2.027976213	10.0.2.15	10.227.107.1…	TLSv1.2	409	Application Data
48	2.028168498	10.227.107.1…	10.0.2.15	TCP	60	443 → 59652 [ACK] Seq=3442 Ack=966 Win=65535 Len=0
49	2.043352862	10.227.107.1…	10.0.2.15	TLSv1.2	87	Encrypted Handshake Message
50	2.043537309	10.0.2.15	10.227.107.1…	TLSv1.2	290	Encrypted Handshake Message
51	2.043751358	10.227.107.1…	10.0.2.15	TCP	60	443 → 59652 [ACK] Seq=3475 Ack=1202 Win=65535 Len=0
52	2.048563004	10.227.107.1…	10.0.2.15	TLSv1.2	3525	Encrypted Handshake Message, Encrypted Handshake Message, Encrypted Handshake Message, Encrypted H
53	2.048579479	10.0.2.15	10.227.107.1…	TCP	54	59652 → 443 [ACK] Seq=1202 Ack=6946 Win=46860 Len=0
54	2.056346271	10.0.2.15	10.227.107.1…	TLSv1.2	202	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message
55	2.056621039	10.227.107.1…	10.0.2.15	TCP	60	443 → 59652 [ACK] Seq=6946 Ack=1350 Win=65535 Len=0
56	2.057116758	10.227.107.1…	10.0.2.15	TLSv1.2	85	Encrypted Alert
57	2.057422989	10.0.2.15	10.227.107.1…	TLSv1.2	85	Encrypted Alert
58	2.057458970	10.227.107.1…	10.0.2.15	TCP	60	443 → 59652 [FIN, ACK] Seq=6977 Ack=1350 Win=65535 Len=0
59	2.057487946	10.0.2.15	10.227.107.1…	TCP	54	59652 → 443 [FIN, ACK] Seq=1381 Ack=6978 Win=46860 Len=0
60	2.057517367	10.227.107.1…	10.0.2.15	TCP	60	[TCP Out-Of-Order] 443 → 59652 [FIN, ACK] Seq=6977 Ack=1381 Win=65535 Len=0
61	2.057535791	10.0.2.15	10.227.107.1…	TCP	54	[TCP Dup ACK 59#1] 59652 → 443 [ACK] Seq=1382 Ack=6978 Win=46860 Len=0
62	2.057565849	10.227.107.1…	10.0.2.15	TCP	60	443 → 59652 [ACK] Seq=6978 Ack=1382 Win=65535 Len=0

Figure 5.5: Network packet analysis: No client certificate presented

3. **Experiment 3**:

   Client's digital certificate and private key bundled into a PKCS12 archive were loaded into *tuxCLI*'s browser.

On *tuxSV1*, the Python script mentioned by the end of section 5.2.3.2 was executed, adding a new entry to a list of entities authorized to access the server. Exact command used (identifying the new identity), a snapshot of the file that contains the list of authorized entities and some information about the used client certificate's subject are expressed in Annex B.3.3.1.

Since information to be matched on *tuxSV1*'s authorized entities list match information in the client certificate, access to the server was successful as expected. Browser requesting which client certificate should be used from the available ones can be seen in figure 5.6.

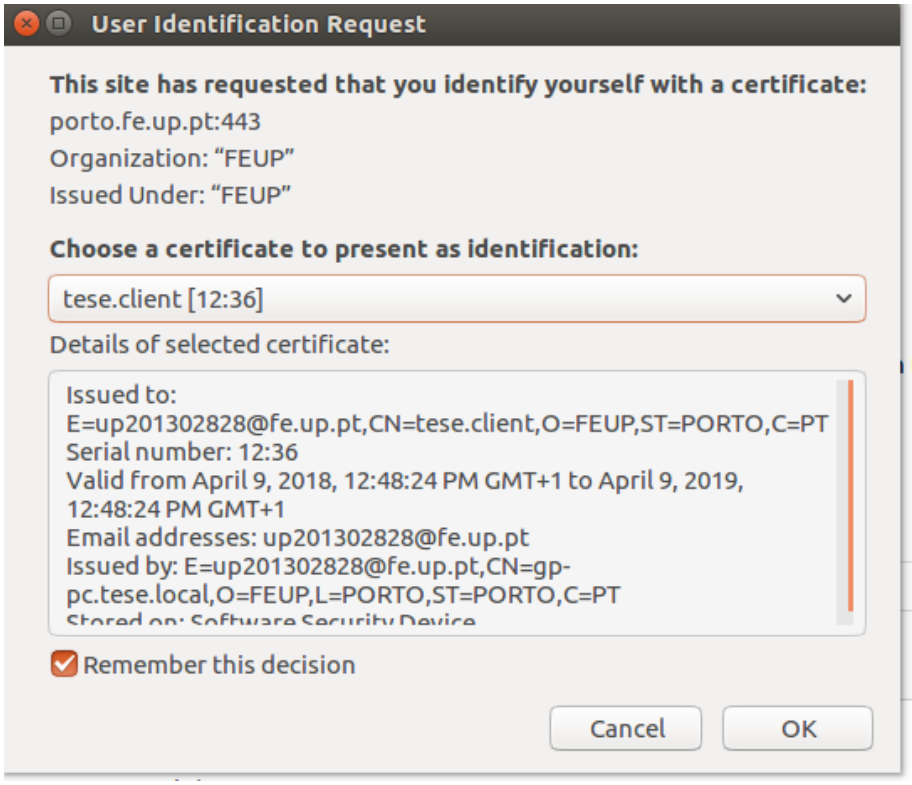


Figure 5.6: Browser requesting client certificate selection

After selection of the correct certificate, successful access through *tuxCLI*'s browser is displayed in figure 5.7.

Annex B.3.3.2 contains network packet analysis, displaying and describing the successful TLS handshake and successful transmission of application data.

4. **Experiment 4**:

   Final experiment's setup is similar to the previous ones and the only preparation needed is removing any entity expression from the servers' authorized entities list that would match the client certificate.

   tuxCLI's browser will once again request a client certificate like figure 5.6, although this time since client certificate's subject information does not match the specified authorized expression, an error message will be displayed on the browser, as seen in figure 5.8.

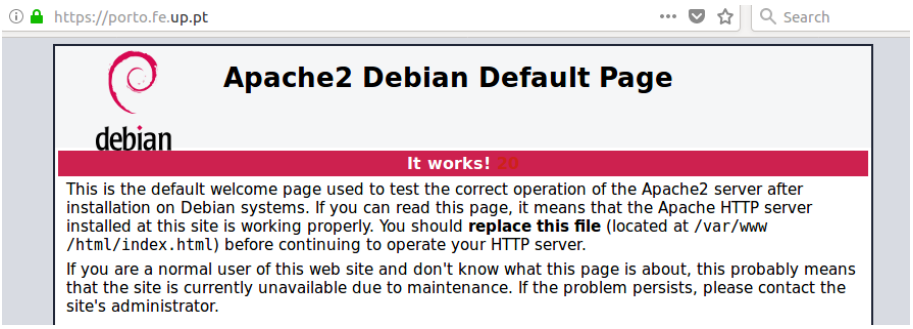


Figure 5.7: Successful access to server's landing page

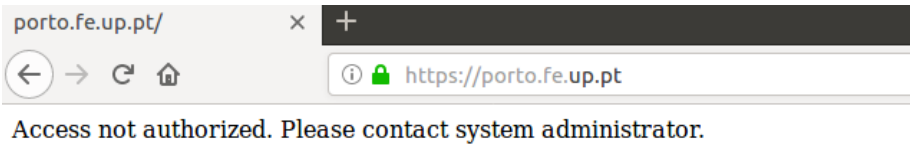


Figure 5.8: Client not authorized to access the server

A final instance of network packets analysis is in Annex B.3.4, that shows *tuxSV1* sending a forbidden response, right after *tuxCLI* sent his certificate, which was deemed an invalid match and therefore not authorized.

5.4 Considerations

Apache server, OpenSSL and TLS process itself are very well tested frameworks and mechanisms and have earned widespread trust between different users, developers and administrators. Solid mechanisms like these require careful implementation and equally important are the versions of the software being used. Existence of updated versions is justified by the discovery of vulnerabilities at a conceptual level targeted at SSL/TLS mechanisms (reviewed in section 2.3.4.1) and implementation flaws in libraries such as OpenSSL. As robustness is a core value of any security process, software must be kept updated to acceptable standards.

Another consideration regarding some cryptography concerns, details such as the size of a private key can and probably will change over time, because often how vulnerable a key is to some attack vectors depends on the capability of contemporaneous computation power and techniques. State of the art on key generations, viability of the algorithms used and other issues should be periodically reviewed to assure the continued security of the system.

Lastly, although this dissertation work used a local CA (CAmed) as a trusted entity by all the peers involved, a real scenario — where reaching a consensus on a fictional trusted CA by all the peers can prove too complex — should require digital certificates to be signed and validated by a professional and specialized entity.

Chapter 6

Data Mining Application Workflow

The ultimate goal of the entire set of processes in this dissertation is allowing for distributed Data Mining (DM) applications to run in a secure manner, within a set of authorized entities. Processes in previous chapters allow for either sharing a (medical) dataset, that before the process contained unacceptable privacy risks, or sharing a data mining model to be used by another peer. Both these shares are done under state-of-the-art secure environments.

Exemplifying possible workflows of DM applications that make use of the architecture developed can add value. Focusing mainly on comparative performances between different scenarios provides concrete measurements on the proportion of maintained effectiveness in a standard use case against utilizing processes developed during the dissertation.

6.1 Machine Learning Algorithms

For a distributed setting, incremental Machine Learning (reviewed in section 2.5) has plenty of potential. Not only it fits a scenario with several peers working simultaneously in the same model or process, but it allows for considerable quantities of data that cannot be stored locally simultaneously to be used as training set. Furthermore, it can adjust to *concept drifting* where the model changes over time to match real world concept altering.

6.1.1 Classification: Linear Support Vector Machine

6.1.1.1 Libraries

Python's package *scikit-learn* provides a classifier that, with correct parameters, behaves as an incremental linear kernel Support Vector Machine (SVM) [1].

Scikit-learn also provides a non-incremental linear SVM[2].

[1] More information at http://scikit-learn.org/stable/modules/sgd.html (Last reviewed in 24 Jun 2018)

[2] Documentation at http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html (Last reviewed in 24 Jun 2018)

Another Python package, *pickle*[3], is usually used to convert an object (such as a DM model) into another format that can be stored, such as a file. This process is known as serialization and the reverse process, deserialization, should also be possible.

Although *pickle* is a common approach, it comes with some security concerns where malicious code can be executed during deserialization. Because of this, only data from trusted sources should be deserialized.

Since in this context only *scikit-learn* models are handled, *joblib* library is used, working similarly as *pickle* but with a better performance (suggested in *scikit-learn* Model Persistence documentation).

As a final note, it is recommended that *scikit-learn* versions match between the users, to avoid any unforeseen complications.

6.1.1.2 Implementation

For training and testing, the Adult dataset was used, both in its original state and the version anonymized by the processes in Chapter 4.

Only conversion from nominal values to numerical values, when relevant, was applied on the dataset preprocessing. Reiterating, the comparison between results in different scenarios is more relevant that the absolute value of the results, so data preprocessing was not extensive.

1. **Non-Incremental SVM**: Unable to load a preexisting model.

   The number of iterations (attempts at generating a model) can be specified, and after preprocessing, performance metrics will be output.

2. **Incremental SVM**: Accepts an argument that specifies if it should generate a new model or if it should load a pre-existing SVM model as default.

   After data preprocessing, training will occur, followed by the output of some performance metrics.

   The number of different models to be generated can also be given as an argument and the best performing one out of the created or incremented models will be stored.

6.1.2 Classification: Online/Incremental Random Forest

Original implementation for this algorithm is present in [Szt17].

This implementation works smoothly with the dataset, even outputting confusion matrices and other performance metrics. A small caveat is that the implementation is of an online algorithm, so while it supports learning from a stream or successive small batches of training data, it is not prepared to store or share the model, nor initiating a new continuation execution after some time, or even in another machine.

To address these issues, adaptations made to the algorithm consist of:

[3]Documentation at `https://docs.python.org/2/library/pickle.html` (Last reviewed in 24 Jun 2018)

1. **Splitting the training set**, allowing for the model to be incrementally trained in different locations / executions of the algorithm, each time with portions of the training data.

2. **Serialization of the generated Random Forest model**, so that the model can be output into a file and shared with other entities, for instance for a new incremental iteration.

3. **Loading of an existing model**, another execution case where a preexisting model is loaded and incremented over.

6.2 Experiments and Results

Metrics used for the results consist mainly of prediction accuracy and confusion matrices.

Confusion matrices cross match the predicted value against the actual value of record. For a quick analysis, it is important to note that frequencies in the main diagonal of the matrix account for the correct predictions, while frequencies outside of it were wrongful predictions.

6.2.1 Non-Incremental SVM

Being non-incremental, the script that trains this model is executed in a single machine, over 100 iterations attempting to find the best model.

6.2.1.1 Original adult dataset

Results obtained by non-incremental SVM with the original dataset are presented in table 6.1. Interpretation of the results will be done further on during this section's conclusions.

Confusion Matrix	Predicted <= 50K	Predicted >50K
Actual <= 50K	3949	4247
Actual >50K	1242	1308

Mean accuracy	0.267
Maximum accuracy	0.489
Minimum accuracy	0.237

Table 6.1: Non-incremental SVM: Original adult dataset

6.2.1.2 Anonymized adult dataset

Table 6.2 contains the output of the best model generated for this scenario, with the anonymized Adult dataset. A relevant note is that not all models generated had zero predictions on '>50K', although coincidentally it happened in the best performing one. Accuracy results variation through all models is small, too; so, results are consistent.

Confusion Matrix	Predicted <= 50K	Predicted >50K
Actual <= 50K	8171	0
Actual >50K	2573	0

Mean accuracy	0.761
Maximum accuracy	0.776
Minimum accuracy	0.750

Table 6.2: Non-incremental SVM: Anonymized adult dataset

6.2.2 Incremental SVM

6.2.2.1 Setup

Dataset to be used was divided into 4 different parts. Firstly, random 30% of the dataset is to be used as test set. The other 70% is the training set, which will be divided again into 3 more or less equally sized parts.

Three different machines will be involved in the process, *tux1* (douro), *tux2* (porto) and *tux3* (gp). All three machines have access to the test set (which are shared by hosting in any of the *tux* machines and the other ones can fetch it securely) and each one of them contains one of the 3 parts that the training set was divided into. Resulting file server on *tux1* is demonstrated in Annexes C.1.1 and C.1.2, for shared datasets and models, respectively.

A *tux* that wants to request a model or dataset from other *tux* uses Python's *requests* library to HTTPS request it, with the correct secure properties, such as digital certificates[4].

6.2.2.2 Original adult dataset

Confusion matrix and accuracy statistics in table 6.3 show the final results in *tux3*, after the model has been incremented by both *tux1* and *tux2* with each of their respective datasets. Intermediate steps can be observed in the Appendix, in section C.2.1.

Confusion Matrix	Predicted <= 50K	Predicted >50K
Actual <= 50K	7095	1101
Actual >50K	2422	128

Mean accuracy	0.310
Maximum accuracy	0.672
Minimum accuracy	0.241

Table 6.3: Incremental SVM: Original adult dataset

[4]Advanced usage of the library available at http://docs.python-requests.org/en/master/user/advanced/ (Last reviewed in 24 Jun 2018)

6.2.2.3 Anonymized adult dataset

Same process is repeated as the original adult dataset for incremental SVM, only using the anonymized dataset, instead. Once more, intermediate results in both *tux1* and *tux2* can be seen in section C.2.2 of the appendix.

Confusion Matrix	Predicted <= 50K	Predicted >50K
Actual <= 50K	7759	412
Actual >50K	2398	175

Mean accuracy	0.551
Maximum accuracy	0.738
Minimum accuracy	0.331

Table 6.4: Incremental SVM: Anonymized adult dataset

6.2.3 Online/Incremental Random Forest

Implementation of Online Random Forest (ORF) has its own dataset, with 8 different possible classes that each record can belong to. The gist of this prediction task is trying to predict the type of motion (climbing up, climbing down, walking, running, sitting, etc) of an individual given some different features. Headers of the dataset and full possible class/label list is given in Annex C.3.1.

Only local and distributed incremental use of the algorithm will be tested, between *tux3* and *tux1*.

6.2.3.1 Local ORF

Results for local execution are displayed in figure 6.1.

```
Correctly Classified Instances        8325               96.9376 %
Incorrectly Classified Instances       263                3.0624 %
Total Number of Instances             8588

FP Rate   Precision   Recall   F-Measure   Class
  0.001       0.995    0.954       0.974   a = climbingdown
  0.010       0.939    0.915       0.927   b = climbingup
  0.000       1.000    1.000       1.000   c = jumping
  0.000       0.999    0.985       0.992   d = lying
  0.007       0.957    0.983       0.970   e = standing
  0.003       0.981    0.994       0.987   f = sitting
  0.000       1.000    0.997       0.999   g = running
  0.014       0.920    0.951       0.935   h = walking
---------------------------------------------------
  0.005       0.970    0.969       0.969   Weighted Avg.

    a     b    c     d     e     f     g     h   <-- classified as
  943    16    0     0    17     0     0    12 |   a = climbingdown
    2  1157    0     0    27     0     0    78 |   b = climbingup
    0     0  150     0     0     0     0     0 |   c = jumping
    0     1    0  1202     6    11     0     0 |   d = lying
    0     1    0     0  1227    10     0    10 |   e = standing
    3     2    0     1     1  1271     0     1 |   f = sitting
    0     0    0     0     0     2  1195     1 |   g = running
    0    55    0     0     4     2     0  1180 |   h = walking
```

Figure 6.1: Local ORF results

6.2.3.2 Distributed incremental ORF

Final results in *tux1* for distributed incremental execution are displayed in figure 6.2.

Results of the first increment performed in *tux3* can be viewed in Annex C.3.2.

```
Successfully loaded model
Serialized data is saved in /tmp/serialized.ser
Correctly Classified Instances        8086               94.1546 %
Incorrectly Classified Instances       502                5.8454 %
Total Number of Instances             8588

FP Rate   Precision   Recall   F-Measure   Class
  0.000       0.997    0.953       0.975   a = climbingdown
  0.011       0.934    0.915       0.924   b = climbingup
  0.000       ◆    0.000       ◆   c = jumping
  0.011       0.937    0.992       0.963   d = lying
  0.007       0.961    0.978       0.969   e = standing
  0.003       0.981    0.937       0.958   f = sitting
  0.023       0.877    0.997       0.933   g = running
  0.013       0.924    0.939       0.931   h = walking
-------------------------------------------------------
  0.010       ◆    0.942       ◆   Weighted Avg.

   a     b  c     d     e     f     g     h   <-- classified as
 942     7  0     0    16     0     8    15 |   a = climbingdown
   0  1157  0     0    27     0     7    73 |   b = climbingup
   0     0  0     0     0     0   150     0 |   c = jumping
   0     0  0  1210     2     8     0     0 |   d = lying
   2     4  0     6  1220     9     0     7 |   e = standing
   1     1  0    76     2  1198     0     1 |   f = sitting
   0     2  0     0     0     2  1194     0 |   g = running
   0    68  0     0     2     4     2  1165 |   h = walking
```

Figure 6.2: Final distributed ORF results

The only difference between a local and distributed ORF application is an intermediate serialization of the *Random Forest* model, its transmission and resuming of the normal process, which should not affect the results obtained. Thus, as expected, differences between results in local and distributed ORF applications are negligible and can be associated with inherent randomness of the algorithm.

6.3 Security of data transmissions

Transmission of the data was monitored the same way as the different scenarios in chapter 4.

Since all three machines were correctly configured, TLS handshakes took place as expected and transmitted data could not be understood without the proper session key, agreed during the handshake.

Annex B.3.5 once more shows the transmission of a data mining model during the process, demonstrating standard TLS handshakes and the model's information unintelligibleness, unless the proper TLS session key is known.

6.4 Conclusions

Regarding a direct comparison between the value of the used datasets with a non incremental SVM, surprisingly the anonymized dataset had superior results overall, by a considerable margin. However, this can possibly be attributed to a lack of correct preprocessing. While the original Adult dataset contains more detailed information, that may also hinder the ability of a classification machine learning algorithm to withdraw generalizations and rules for records. So, even though reasonable deviations were expected, the underachieving results of a linear SVM with the original dataset are, most likely, an outlier that could be corrected with further feature engineering and data preprocessing.

On comparing incremental and non-incremental SVM's performances, the results seem quite similar, over some iterations. The minimum and mean accuracies of these incremental approaches show that they can be less reliable. Nevertheless, since incremental SVM supports learning over small quantities of data at a time, while standard SVM must be trained with a single batch, it is possible that the former can afford to try more iterations to find a good classifier than the latter.

The second algorithm used, ORF, presents very similar results when used locally or distributively, with negligible variations associated with inherent randomization in the ORF algorithm. Adaptations to include serialization and deserialization processes seem to have been successful.

Data transmission between entities was also made under the proposed secure processes in this dissertation, allowing for the complete application workflow to be kept exclusively within the three peers involved.

6.5 Considerations

Some research on the privacy dangers that may exist when sharing data mining models and accesses to models has been made. For instance, [KJC04] presents some scenarios where the output of a DM model could violate privacy, including the most straightforward scenario where public data can be input into the model and the output is a sensitive attribute about the individual. Although reverse engineering the utilized dataset from a model may be unfeasible, full access to models and to some partial information about some individuals may lead to privacy leaks.

Given how this behavior can change depending on the model being shared, it is hard to generalize rules. Best course of action should be to maintain access to these distributed DM applications between trusted entities. The developed architecture allows for managing of the entities that can partake in an application and to guarantee maximum privacy, but this access must be handled with careful selection.

Processes in the dissertation allow for secure channels of communication and privacy-preserving methods of dataset sharing. Even so, not only should the entities that can access resources be controlled, the entities themselves must follow secure practices to avoid situations of data theft, social engineering or exploits that may grant, for instance, unauthorized accesses to their own machines.

Chapter 7

Conclusions and Future Work

This dissertation work has a main goal of tackling security concerns in the area of medical Data Mining applications. These concerns gain importance as the area expands with the amount of data available and the number of entities involved.

Distributed medical Data Mining applications thus become a powerful method to handle this expanding context. With a mostly distributed setting, secure channels must be created to contain and transmit sensitive (here, medical) information between authorized entities. Also, anonymization of datasets allows more entities to handle the information, supporting further research in a possibly life-saving area.

Incremental Data Mining applications also provide means for different entities to cooperate in a single Data Mining application and improve the value of a Data Mining model, simultaneously benefiting all peers involved.

Data anonymization was developed while complying to acceptable re-identification risk thresholds for medical data. The process developed was intentionally organic to provide a deep understanding of the decisions to be made. Results in organic anonymization measured by the loss of information were acceptable or good on most of the attributes, while still leaving room for more experienced intermediate decisions or generalization hierarchies that could wield better results.

Secure transmission of data was implemented according to state-of-the-art mechanisms. This implementation features an intuitively organized distribution of files in a server and the ability to efficiently grant new entities access to the files on a server. As long as an entity can provide proper authentication and is authorized through the proposed mechanisms, access to the data should be possible and security assured during data transmission.

Incremental Data Mining algorithms were expanded on, since most open-source implementations available do not allow for transmission of the model generated. Prototypical implementations of Data Mining applications revealed that the proposed distributed architecture can securely and conveniently transmit data with minimal or non existent impact on the performance of the algorithms. Further preprocessing of the utilized datasets could increase the reliability of the results extracted, by simulating a more realistic Data Mining application.

Some crucial concerns have been presented for the maintenance of the secure properties of the architecture, and must be abided.

Overall, the proposed solution assures a privacy-preserving and secure environment to develop further research on medical Data Mining applications.

As the initial goals have been met, the only foreseeable future work lies on the automation of data anonymization by phrasing it as an optimization problem, exclusively for convenience purposes. All other future work is left for researchers in distributed medical Data Mining.

References

[AA13] Mohammad A AlAhmad and Imad Fakhri Alshaikhli. Broad view of cryptographic hash functions. *International Journal of Computer Science Issues*, 10(4):239–246, 2013.

[AG07] Ashraf Abusharekh and Kris Gaj. Comparative analysis of software libraries for public key cryptography. *Software Performance Enhancement for Encryption and Decryption, SPEED*, pages 11–12, 2007.

[AH+12] Fareed Akthar, Caroline Hahne, et al. Rapidminer 5. *Operator Reference. www. rapid-i. com*, 2012.

[CGI04] CGI. Public key encryption and digital signature: How do they work? Available at https://www.cgi.com/files/white-papers/cgi_whpr_35_pki_e.pdf, last reviewed in 19 Jan 2018, 2004.

[CKLM09] Bee-Chung Chen, Daniel Kifer, Kristen LeFevre, and Ashwin Machanavajjhala. Privacy-preserving data publishing. *Found. Trends databases*, 2(1–2):1–167, January 2009.

[DC03] Christopher P Diehl and Gert Cauwenberghs. Svm incremental learning, adaptation and optimization. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 4, pages 2685–2690. IEEE, 2003.

[DHV03] Janaka Deepakumara, Howard M Heys, and R Venkatesan. Performance comparison of message authentication code (mac) algorithms for internet protocol security (ipsec). In *Proc. Newfoundland Electrical and Computer Engineering Conf*, 2003.

[DKT17] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository. Available at https://archive.ics.uci.edu/ml/datasets/adult, last reviewed in 10 Jun 2018, 2017.

[Dob09] Alin Dobra. *Decision Tree Classification*, pages 765–769. Springer US, Boston, MA, 2009.

[DR08] T. Dierks and E. Rescorla. The transport layer security (TLS) protocol version 1.2. Technical report, August 2008.

[EE11] Khaled El Emam. Methods for the de-identification of electronic health records for genomic research. *Genome medicine*, 3(4):25, 2011.

[EED08] K. El Emam and F.K. Dankar. Protecting privacy using k-anonymity. *Journal of the American Medical Informatics Association*, 15(5):627–637, 2008.

[EERM15] Khaled El Emam, Sam Rodgers, and Bradley Malin. Anonymising and sharing individual patient data. *bmj*, 350:h1139, 2015.

[ESM+15] Abeer EW Eldewahi, Tasneem MH Sharfi, Abdelhamid A Mansor, Nashwa AF Mohamed, and Samah MH Alwahbani. SSL/TLS attacks: Analysis and evaluation. In *Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE), 2015 International Conference on*, pages 203–208. IEEE, 2015.

[Fis36] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of human genetics*, 7(2):179–188, 1936.

[Gra15] Laura K. Gray. Date change for migrating from SSL and early TLS. Available at https://blog.pcisecuritystandards.org/migrating-from-ssl-and-early-tls, last reviewed in 28 Jan 2018, December 2015.

[Hel02] Martin E Hellman. An overview of public key cryptography. *IEEE Communications Magazine*, 40(5):42–49, 2002.

[Her10] P Herzog. Osstmm 3–the open source security testing methodology manual. last reviewed in 07 Feb 2018 at http://www.isecom.org/mirror/OSSTMM.3.pdf, 2010.

[HFPS08] Russell Housley, Warwick Ford, W Polk, and David Solo. Rfc 5280: Internet x. 509 public key infrastructure certificate and crl profile, 2008.

[HMS01] David J Hand, Heikki Mannila, and Padhraic Smyth. *Principles of data mining (adaptive computation and machine learning)*. MIT press Cambridge, MA, 2001.

[JD88] Anil K Jain and Richard C Dubes. Algorithms for clustering data. 1988.

[KCB97] Hugo Krawczyk, Ran Canetti, and Mihir Bellare. Hmac: Keyed-hashing for message authentication. 1997.

[KGNK16] Venkata SreeKrishna Koganti, Lavanya K Galla, Nagarjuna Nuthalapati, and Anil Varma Kakarlapudi. Authentication protocols using encryption techniques. In *Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2016 International Conference on*, pages 74–80. IEEE, 2016.

[KJC04] Murat Kantarcioğlu, Jiashun Jin, and Chris Clifton. When do data mining results violate privacy? In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 599–604. ACM, 2004.

[KMN+02] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):881–892, 2002.

[Koh96] Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In *KDD*, volume 96, pages 202–207. Citeseer, 1996.

[KS14] J. Udhayan A K. Saranya, R. Mohanapriya. Review on symmetric key encryption techniques in cryptography. *International Journal of Science, Engineering and Technology Research (IJSETR), Volume 3, Issue 3,*, March 2014.

REFERENCES

[KT12] Batya Kenig and Tamir Tassa. A practical approximation algorithm for optimal k-anonymity. *Data Mining and Knowledge Discovery*, 25(1):134–168, 2012.

[LHW18] Viktor Losing, Barbara Hammer, and Heiko Wersing. Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing*, 275:1261–1274, 2018.

[MM14] Matteo Meucci and A Muller. Owasp testing guide. *v4. 0,//OWASP Foundation*, (s 453), 2014.

[MT14] Ltd. Milieu and Time.lex. Overview of the national laws on electronic health records in the EU member states and their interaction with the provision of cross-border ehealth services. Available at https://ec.europa.eu/digital-single-market/en/news/study-national-laws-electronic-health-records-and-cross-border-ehealth-services-eu-member, 2014.

[Pit03] Steve Pitts. An overview of digital certificates and how they are used in vpn authentication. Available at https://www.giac.org/paper/gsec/2711/overview-digital-certificates-vpn-authentication/104625 ; Last reviewed in 05 Feb 2018, March 2003.

[PNNM16] Priyadarshini Patil, Prashant Narayankar, DG Narayan, and SM Meena. A comprehensive evaluation of cryptographic algorithms: DES, 3DES, AES, RSA and blowfish. *Procedia Computer Science*, 78:617–624, 2016.

[PVG+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[Res05] Eric Rescorla. Rfc 2818: Http over tls, may 2000. *Available from World Wide Web: ftp://ftp. rfc-editor. org/in-notes/rfc2818. txt. Status: Informational*, 2005.

[SS98] Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, Technical report, SRI International, 1998.

[Szt17] Timo Sztyler. Online random forest classifier (java). https://github.com/sztyler/online-random-forest, 2017.

[VMC02] John Viega, Matt Messier, and Pravir Chandra. *Network Security with OpenSSL: Cryptography for Secure Communications*. " O'Reilly Media, Inc.", 2002.

[WH00] Rüdiger Wirth and Jochen Hipp. Crisp-dm: Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, pages 29–39. Citeseer, 2000.

[WY05] Xiaoyun Wang and Hongbo Yu. How to break md5 and other hash functions. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 19–35. Springer, 2005.

[Z+16] Yu Z et al. Incremental semi-supervised clustering ensemble for high dimensional data clustering. *IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 3*, pages 701–714, March 2016.

REFERENCES

Appendix A

Scripts output

A.1 Execution times

A.1.1 Normal iteration with tabu dictionary

Here the script that is described in Algorithm 1 of section 4.3.3 was executed until the 1105th row out of 32561 rows, using the original Adult dataset, before any generalization or alteration was made.

```
> time python anon_study.py

0
1
2
3
4
5

[Ommited index printing]


1096
1097
1098
1099
1100
1101
1102
1103
1104
```

```
1105
^CTraceback (most recent call last):
  File "anon_study.py", line 37, in <module>
    for secondary_index in range(current_index+1,complete_list_size):
KeyboardInterrupt

real 10m46.324s
user 10m46.252s
sys 0m0.512s
```

A.1.2 Set implementation

The output of the script whose algorithm is described in Algorithm 2 of section 4.3.3, over the unaltered Adult dataset.

```
time python anon_study_set.py
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
5500
6000
6500
7000
7500
8000
8500
9000

[Ommited index printing]

30000
30500
```

```
31000
31500
32000
32500
Main iteration execution time: 9.13195610046
set-union size: 374191
set-intersection size: 77182

real 0m11.158s
user 0m10.452s
sys 0m0.516s
```

A.2 Iterative anonymization analysis outputs

All outputs will follow the format of the example set by Annex A.2.1. Portion of the output can be omitted, so that the presented information focuses on the data necessary to support the decision for the actions to take on each ongoing iteration.

A.2.1 Example analysis during anonymization process

The script outputs, for the set of pruned equivalence classes, the size of the smallest equivalence class ("smallest size"), the number of occurrences of each attribute in these problematic equivalence classes, counters of the occurrence of each value and total number of problematic combinations ("smallest_only_list size").

```
smallest size: 2
age occurrences: 24
education occurrences: 22
marital status occurrences: 21
occupation occurrences: 7
race occurrences: 18
sex occurrences: 20
native_country occurrences: 23

 COUNTERS :::
Counter({'60': 12, '20': 8, '40': 4})
Counter({'Basic': 11, 'Superior': 6, 'HighSchool': 5})
Counter({'Married-before': 10, 'Never-married': 6, 'Married': 5})
```

```
Counter({'adm_manag': 4, 'tech_labour': 2, 'care_oriented': 1})
Counter({'Other': 14, 'Black': 2, 'Amer-Indian-Eskimo': 1, 'White': 1})
Counter({'Asia': 12, 'America': 7, 'Europe': 4})

smallest_only_list size: 36
```

A.2.2 Iteration 2 analysis

```
smallest size: 2
age occurrences: 304
education occurrences: 348
marital status occurrences: 126
occupation occurrences: 317
race occurrences: 74
sex occurrences: 0
native_country occurrences: 934

Counter({'Trinadad&Tobago': 33, 'Honduras': 32, 'Nicaragua': 32,
'Ecuador': 31, 'Hong': 30, 'Peru': 29, 'Thailand': 29, 'Greece': 29,
'Vietnam': 28, 'Yugoslavia': 28, 'Laos': 27, 'Columbia': 27, 'Poland': 27,
'Scotland': 26, 'Cambodia': 26, 'France': 26, 'Haiti': 26, 'Ireland': 25,
'Hungary': 25, 'Portugal': 25, 'Italy': 24, 'Outlying-US(Guam-USVI-etc)': 24,
'England': 23, 'Cuba': 22, 'Canada': 21, 'Guatemala': 21, 'Iran': 21,
'El-Salvador': 21, 'Dominican-Republic': 19, 'Germany': 19, 'Jamaica': 19,
'Japan': 19, 'Taiwan': 19, 'China': 18, 'South': 18, 'Philippines': 16,
'India': 16, 'Puerto-Rico': 16, 'Mexico': 10, 'Holand-Netherlands': 7})
```

A.2.3 Iteration 3 analysis

```
age occurrences: 109
education occurrences: 99
marital status occurrences: 40
occupation occurrences: 89
race occurrences: 33
sex occurrences: 0
```

```
native_country occurrences: 0
```

A.2.4 Iteration 3.1 analysis

```
age occurrences: 43
education occurrences: 76
marital status occurrences: 32
occupation occurrences: 67
race occurrences: 23
sex occurrences: 0
native_country occurrences: 0
```

A.2.5 Iteration 4 analysis

```
smallest size: 2
age occurrences: 65
education occurrences: 0
marital status occurrences: 29
occupation occurrences: 49
race occurrences: 26
sex occurrences: 0
native_country occurrences: 0
```

A.2.6 Iteration 4.1 analysis

```
smallest size: 2
age occurrences: 26
education occurrences: 0
marital status occurrences: 21
occupation occurrences: 27
race occurrences: 16
sex occurrences: 0
native_country occurrences: 0
```

A.2.7 Iteration 5 analysis

```
age occurrences: 8
education occurrences: 0
marital status occurrences: 18
occupation occurrences: 20
race occurrences: 13
sex occurrences: 0
native_country occurrences: 0

>>> Counter(age_vals)
Counter({'20': 3, '50': 3, '60': 1, '40': 1})
>>> Counter(marital_status_vals)
Counter({'Married-AF-spouse': 13, 'Widowed': 2, 'Married-spouse-absent': 2,
'Married-civ-spouse': 1})
>>> Counter(occupation_vals)
Counter({'Armed-Forces': 6, 'Priv-house-serv': 4, 'Tech-support': 2,
'Farming-fishing': 1, 'Craft-repair': 1, 'Protective-serv': 1,
'Transport-moving': 1, 'Prof-specialty': 1, 'Sales': 1, 'Exec-managerial': 1,
'Other-service': 1})
>>> Counter(race_vals)
Counter({'Amer-Indian-Eskimo': 4, 'Other': 4, 'Black': 3,
'Asian-Pac-Islander': 2})
```

A.2.8 Iteration 7 analysis

```
smallest size: 3
age occurrences: 240
education occurrences: 0
marital status occurrences: 245
occupation occurrences: 433
race occurrences: 289
sex occurrences: 0
native_country occurrences: 0
```

A.2.9 Iteration 8 analysis

```
smallest size: 3
age occurrences: 80
education occurrences: 40
marital status occurrences: 103
occupation occurrences: 36
race occurrences: 116
sex occurrences: 0
native_country occurrences: 79
```

A.2.10 Iteration 9 analysis

```
smallest size: 3
age occurrences: 1024
education occurrences: 852
marital status occurrences: 888
occupation occurrences: 801
race occurrences: 951
sex occurrences: 0
native_country occurrences: 683
 COUNTERS :::
Counter({'40': 246, '60': 232, '30': 227, '50': 217, '20': 102})
Counter({'Married-before': 340, 'Never-married': 280, 'Married': 268})
Counter({'care_oriented': 282, 'tech_labour': 265, 'adm_manag': 254})
Counter({'Asian-Pac-Islander': 348, 'Other': 180, 'Black': 153,
'Amer-Indian-Eskimo': 140, 'White': 130})

total number_of_combinations:
1293
```

A.2.11 Iteration 10 analysis

```
smallest size: 3
age occurrences: 410
education occurrences: 439
marital status occurrences: 446
occupation occurrences: 497
race occurrences: 442
sex occurrences: 0
```

```
native_country occurrences: 391
 COUNTERS :::
Counter({'60': 171, '40': 140, '20': 99})
Counter({'Superior': 162, 'Basic': 157, 'HighSchool': 120})
Counter({'Married-before': 163, 'Never-married': 146, 'Married': 137})
Counter({'tech_labour': 137, 'care_oriented': 130, 'adm_manag': 121,
'manual': 109})
Counter({'Other': 211, 'Amer-Indian-Eskimo': 82, 'Black': 75, 'White': 74})
Counter({'Europe': 180, 'Asia': 141, 'America': 70})

total number_of_combinations:
621
```

A.3 Server security related

A.3.1 Usage help on adding new authorized entity

```
> python add_new_accepted_entity.py -h
usage: add_new_accepted_entity.py [-h] [-c COUNTRY] [-st STATE] [-l LOCATION]
                                  [-o ORGANIZATION] [-ou ORGANIZATION_UNIT]
                                  [-cn COMMON_NAME] [-cf CONFIG_FILE]

Add new accepted client to access the shared models or datasets, through its
certificate's properties

optional arguments:
  -h, --help            show this help message and exit
  -c COUNTRY            Country abbreviation
  -st STATE             State
  -l LOCATION           Locality
  -o ORGANIZATION       Organization
  -ou ORGANIZATION_UNIT
                        Organization Unit
  -cn COMMON_NAME       Common Name
  -cf CONFIG_FILE       Configuration file included in apache SSL
                        configuration (default:
                        /etc/apache2/valid_expressions.list)
```

Appendix B

Security Configuration and Implementation

B.1 Configuration files

B.1.1 Hosts file

Normally located in /etc/hosts, during the creation of the Certificate Authority, it was edited to contain the following:

```
127.0.0.1       localhost
127.0.1.1       gp-pc.tese.local GP-PC
```

where 'gp-pc.tese.local' is the Fully Qualified Domain Name and GP-PC is the simple hostname.

B.1.2 OpenSSL configuration file

OpenSSL's configuration can be altered in a file named openssl.cnf, usually located in /usr/lib/ssl.

Under section [CA_default] of the file, the value 'dir' should be set to the root directory of where Certificate Authority's files are to be stored, like such:

```
[ CA_default ]

dir = /root/ca
```

Furthermore, default policy should be looked at on the [policy_match] section, and should be adjusted to each use case. Some more information can be obtained here: https://www.phildev.net/ssl/opensslconf.html and on OpenSSL's documentation.

B.1.3 .htaccess file - forcing HTTPS connection

```
  GNU nano 2.2.6                                        File: .htaccess

RewriteEngine On

RewriteCond %{SERVER_PORT} 80

RewriteRule ^(.*)$ https://porto.fe.up.pt/$1 [R=301,L]

RewriteCond     %{SSL:SSL_CLIENT_VERIFY} !=SUCCESS
RewriteRule     .? - [F]
ErrorDocument 403 "You need a client side certificate issued by CAmed
to access this site or you not authorized to access this page.
If the second, please contact system administrator."
```

where 'porto.fe.up.pt' in 'https://porto.fe.up.pt/$1' must be replaced by the server's address.

B.1.4 Authorized entities validation file

```
SSLRequire ( %{SSL_CLIENT_S_DN_O}  eq "FEUP2" \
and %{SSL_CLIENT_S_DN_CN} in {"tese.clientaa", "CaaA", "Deaav"}) \
or (%{SSL_CLIENT_S_DN_ST} eq "PORTO" and %{SSL_CLIENT_S_DN_CN} in
{"tese.asdfe", "CaaA"}) \
or (%{SSL_CLIENT_S_DN_O} eq "FEUP" and %{SSL_CLIENT_S_DN_CN}
eq "tese.clienttt") \
or (%{SSL_CLIENT_S_DN_O} eq "FEUP2" and %{SSL_CLIENT_S_DN_CN}
eq "tese.client2") \
or (%{SSL_CLIENT_S_DN_O} eq "FEUP3" and %{SSL_CLIENT_S_DN_CN}
eq "tese.client32") \
or (%{SSL_CLIENT_S_DN_O} eq "FEUP3asd" \
and %{SSL_CLIENT_S_DN_CN} eq "tese.client32") \
or (%{SSL_CLIENT_S_DN_C} eq "NL" and \
%{SSL_CLIENT_S_DN_CN} eq "nether.lands") \
or (%{SSL_CLIENT_S_DN_C} eq "NL" and %{SSL_CLIENT_S_DN_CN}
eq "nether.lands")or (%{SSL_CLIENT_S_DN_C} eq "NL" \
and %{SSL_CLIENT_S_DN_CN} eq "nether.lands") \
or (%{SSL_CLIENT_S_DN_C} eq "PT" and \
%{SSL_CLIENT_S_DN_O} eq "FEUP" \
and %{SSL_CLIENT_S_DN_CN} eq "tese.client")
```

B.2 Certificate and key generations

B.2.1 Root certificate

```
> openssl req -new -x509 -key private/cakey.pem \
-out certs/cacert.pem -days 3650 -set_serial 0

Enter pass phrase for /root/ca/private/cakey.pem:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:PT
State or Province Name (full name) [Some-State]:PORTO
Locality Name (eg, city) []:PORTO
Organization Name (eg, company) [Internet Widgits Pty Ltd]:FEUP
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:gp-pc.tese.local
Email Address []:up201302828@fe.up.pt
```

Common Name inserted matches the fully qualified domain name, that can be checked by inputting the terminal command:

```
> hostname -f
```

B.2.2 SubjectAltName compliance

[Res05] specifies that SubjectAltName (SAN) field must be present in some situations, even though following of this standard is not too often enforced. Nonetheless, browsers like Chrome on its version 58 has begun to require SAN field in digital certificates. Standard methods of generating a *Certificate Signing Request* (CSR) are usually not sufficient to generate this field. To do so, a configuration file may be created as below:

```
[req]
default_bits = 2048
```

```
prompt = no
default_md = sha256
req_extensions = req_ext
distinguished_name = dn

[ dn ]
C=PT
ST=PORTO
L=PORTO
O=FEUP
OU=DISS
emailAddress=gustavo.pinto@fe.up.pt
CN = porto.fe.up.pt

[ req_ext ]
subjectAltName = @alt_names

[ alt_names ]
DNS.1 = porto.fe.up.pt
DNS.2 = www.porto.fe.up.pt
```

When stored on a file named "csr_details.txt", creation of a CSR would now feature the following command:

```
> openssl req -new -key peer_private_key.pem -out peer_req.csr \
-config <( cat csr_details.txt )
```

B.2.3 Signing of a CSR

```
> openssl ca -in peer_req.csr -out peer_cert.pem

Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for /root/ca/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4668 (0x123c)
```

```
        Validity
            Not Before: Jun 19 00:46:08 2018 GMT
            Not After : Jun 19 00:46:08 2019 GMT
        Subject:
            countryName               = PT
            stateOrProvinceName       = PORTO
            organizationName          = FEUP
            organizationalUnitName    = DISS
            commonName                = porto.fe.up.pt
            emailAddress              = gustavo.pinto@fe.up.pt
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                1C:6A:78:CF:C0:F6:52:7C:86:E8:00:EE:A2:78:55
                :D8:80:DB:51:82
            X509v3 Authority Key Identifier:
                keyid:59:0B:A0:31:20:50:BD:DE:6A:
                0E:21:8A:C1:50:A2:FA:F9:83:C5:B9

Certificate is to be certified until Jun 19 00:46:08 2019 GMT (365 days)
Sign the certificate? [y/n]:
```

On signing of a CSR — since CA's password was created with password protection — the password for '/root/ca/private/cakey.pem' is requested, followed by a verification that the CSR has not been corrupted, by checking its digital signature.

Several details are then displayed, such as tentative serial number, validity, information inserted by the requester that claims its identity (country, organization name, common name, etc.), as well as some technical details such as the version of X509, the standard file format for digital certificates.

Finally, a confirmation prompt to sign the certificate is displayed. If signed, another confirmation prompt to commit the newly generated certificate in CA's database may appear:

```
1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

B.2.4 Generating PKCS archive

Once again, OpenSSL provides the necessary commands to convert a private key and a digital certificate into a single bundle, as such:

```
> openssl pkcs12 -export -inkey peer_private_key.pem  -in peer_cert.pem \
-out client_p12.p12

Enter pass phrase for peer_private_key.pem:
Enter Export Password:
Verifying - Enter Export Password:
```

This command shall output a p12 (since PKCS12 was used) archive that can be imported, for example, into a browser.

As usual, access to a private key generated with AES password encryption will request its password on access.

Furthermore, since the PKCS12 archive contains again sensitive information, a password must also be used to prevent unauthorized use or access to the bundle, hence why the last 2 prompts request an "Export password".

B.3 Experiments output

B.3.1 TLS version verification

Both nmap and OpenSSL tools can be used to verify available TLS versions. [1]

Nmap's output shows all the available ciphers under each TLS version in a file tree format:

```
> nmap --script ssl-enum-ciphers -p 443 porto.fe.up.pt

Starting Nmap 7.01 ( https://nmap.org ) at 2018-06-20 03:49 WEST
Nmap scan report for porto.fe.up.pt (192.168.101.172)
Host is up (0.00071s latency).
PORT    STATE SERVICE
443/tcp open  https
```

[1] TLSv1.0 was still left as legacy to allow for support of outdated clients, but as stated often throughout the dissertation, it is deprecated and no longer deemed secure.

```
| ssl-enum-ciphers:
|   TLSv1.0:
|     ciphers:
|       TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (dh 2048) - C
|       TLS_DHE_RSA_WITH_AES_128_CBC_SHA (dh 2048) - A
|       TLS_DHE_RSA_WITH_AES_256_CBC_SHA (dh 2048) - A
|       TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA (dh 2048) - A
|       TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (dh 2048) - A
|       TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA (secp256r1) - C
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp256r1) - A
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA (rsa 2048) - C
|       TLS_RSA_WITH_AES_128_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_CAMELLIA_128_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_CAMELLIA_256_CBC_SHA (rsa 2048) - A
|     compressors:
|       NULL
|     cipher preference: client
|   TLSv1.1:
|     ciphers:
|       TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (dh 2048) - C
|       TLS_DHE_RSA_WITH_AES_128_CBC_SHA (dh 2048) - A
|       TLS_DHE_RSA_WITH_AES_256_CBC_SHA (dh 2048) - A
|       TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA (dh 2048) - A
|       TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (dh 2048) - A
|       TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA (secp256r1) - C
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp256r1) - A
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA (rsa 2048) - C
|       TLS_RSA_WITH_AES_128_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_CAMELLIA_128_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_CAMELLIA_256_CBC_SHA (rsa 2048) - A
|     compressors:
|       NULL
|     cipher preference: client
|   TLSv1.2:
|     ciphers:
|       TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (dh 2048) - C
```

```
|       TLS_DHE_RSA_WITH_AES_128_CBC_SHA (dh 2048) - A
|       TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 (dh 2048) - A
|       TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (dh 2048) - A
|       TLS_DHE_RSA_WITH_AES_256_CBC_SHA (dh 2048) - A
|       TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (dh 2048) - A
|       TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (dh 2048) - A
|       TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA (dh 2048) - A
|       TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA (dh 2048) - A
|       TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA (secp256r1) - C
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (secp256r1) - A
|       TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (secp256r1) - A
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA (rsa 2048) - C
|       TLS_RSA_WITH_AES_128_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_AES_128_CBC_SHA256 (rsa 2048) - A
|       TLS_RSA_WITH_AES_128_GCM_SHA256 (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_CBC_SHA256 (rsa 2048) - A
|       TLS_RSA_WITH_AES_256_GCM_SHA384 (rsa 2048) - A
|       TLS_RSA_WITH_CAMELLIA_128_CBC_SHA (rsa 2048) - A
|       TLS_RSA_WITH_CAMELLIA_256_CBC_SHA (rsa 2048) - A
|     compressors:
|       NULL
|     cipher preference: client
|_  least strength: C

Nmap done: 1 IP address (1 host up) scanned in 0.60 seconds
```

Meanwhile, OpenSSL s_client allows simulation of a client on the server, while forcing the usage of a specific TLS version. The absence of error messages and a valid created SSL-session demonstrates the server's support.

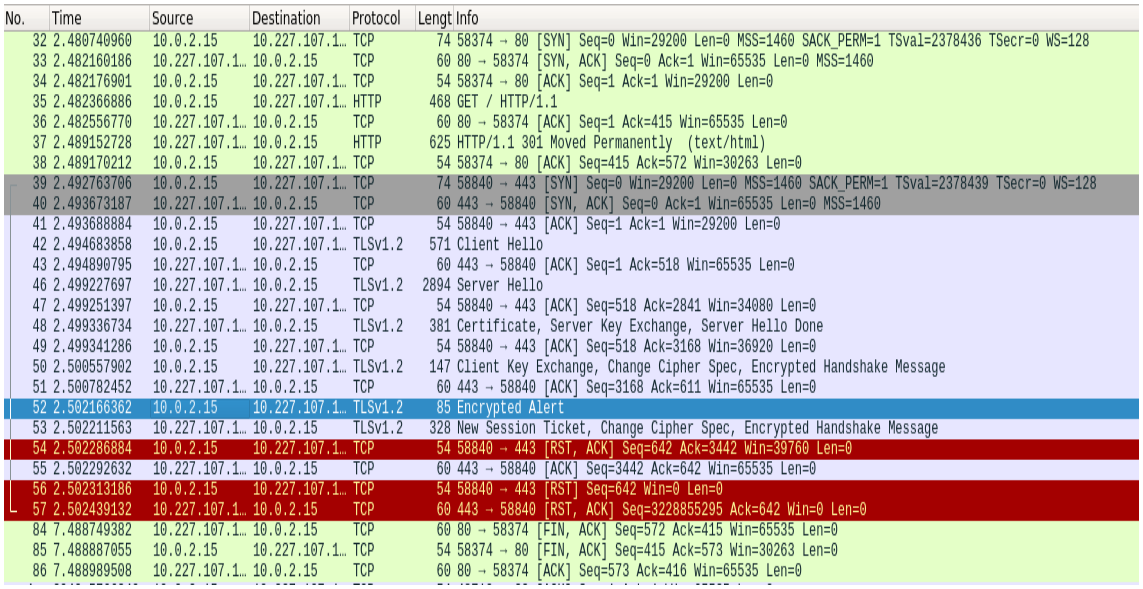
```
> openssl s_client -connect porto.fe.up.pt:443 -tls1_2


CONNECTED(00000003)
```

```
depth=1 C = PT, ST = PORTO, L = PORTO, O = FEUP,
CN = gp-pc.tese.local, emailAddress = up201302828@fe.up.pt
verify error:num=19:self signed certificate in certificate chain
---
Certificate chain
 0 s:/C=PT/ST=PORTO/L=PORTO/O=FEUP/OU=DISS/CN=porto.fe.up.pt/
 emailAddress=gustavo.pinto@fe.up.pt
   i:/C=PT/ST=PORTO/L=PORTO/O=FEUP/CN=gp-pc.tese.local/
   emailAddress=up201302828@fe.up.pt
 1 s:/C=PT/ST=PORTO/L=PORTO/O=FEUP/CN=gp-pc.tese.local/
 emailAddress=up201302828@fe.up.pt
   i:/C=PT/ST=PORTO/L=PORTO/O=FEUP/CN=gp-pc.tese.local/
   emailAddress=up201302828@fe.up.pt
---
Server certificate
-----BEGIN CERTIFICATE-----
<truncated information>
-----END CERTIFICATE-----
subject=/C=PT/ST=PORTO/L=PORTO/O=FEUP/OU=DISS/CN=porto.fe.up.pt/
emailAddress=gustavo.pinto@fe.up.pt
issuer=/C=PT/ST=PORTO/L=PORTO/O=FEUP/CN=gp-pc.tese.local/
emailAddress=up201302828@fe.up.pt
---
No client certificate CA names sent
Peer signing digest: SHA512
Server Temp Key: ECDH, P-256, 256 bits
---
SSL handshake has read 3406 bytes and written 431 bytes
---

<truncated information>

SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES256-GCM-SHA384

    <truncated information>

    Start Time: 1529463133
    Timeout   : 7200 (sec)
```

```
    Verify return code: 19 (self signed certificate in certificate chain)
---
```

B.3.2 Experimental scenario 1

Figure B.1 shows TLS communications going smoothly, until packet number 52, where an encrypted alert is sent from the client *tuxCLI* to the server *tuxSV2*, cancelling the communications, followed by TCP closing messages [FIN]. This behavior is justified by *tuxCLI* rejecting the certificate sent by *tuxSV2* in packet number 48.



No.	Time	Source	Destination	Protocol	Lengt	Info
32	2.480740960	10.0.2.15	10.227.107.1…	TCP	74	58374 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2378436 TSecr=0 WS=128
33	2.482160186	10.227.107.1…	10.0.2.15	TCP	60	80 → 58374 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
34	2.482176901	10.0.2.15	10.227.107.1…	TCP	54	58374 → 80 [ACK] Seq=1 Ack=1 Win=29200 Len=0
35	2.482366886	10.0.2.15	10.227.107.1…	HTTP	468	GET / HTTP/1.1
36	2.482556770	10.227.107.1…	10.0.2.15	TCP	60	80 → 58374 [ACK] Seq=1 Ack=415 Win=65535 Len=0
37	2.489152728	10.227.107.1…	10.0.2.15	HTTP	625	HTTP/1.1 301 Moved Permanently (text/html)
38	2.489170212	10.0.2.15	10.227.107.1…	TCP	54	58374 → 80 [ACK] Seq=415 Ack=572 Win=30263 Len=0
39	2.492763706	10.0.2.15	10.227.107.1…	TCP	74	58840 → 443 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=2378439 TSecr=0 WS=128
40	2.493673187	10.227.107.1…	10.0.2.15	TCP	60	443 → 58840 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
41	2.493688884	10.0.2.15	10.227.107.1…	TCP	54	58840 → 443 [ACK] Seq=1 Ack=1 Win=29200 Len=0
42	2.494683858	10.0.2.15	10.227.107.1…	TLSv1.2	571	Client Hello
43	2.494890795	10.227.107.1…	10.0.2.15	TCP	60	443 → 58840 [ACK] Seq=1 Ack=518 Win=65535 Len=0
46	2.499227697	10.227.107.1…	10.0.2.15	TLSv1.2	2894	Server Hello
47	2.499251397	10.0.2.15	10.227.107.1…	TCP	54	58840 → 443 [ACK] Seq=518 Ack=2841 Win=34080 Len=0
48	2.499336734	10.227.107.1…	10.0.2.15	TLSv1.2	381	Certificate, Server Key Exchange, Server Hello Done
49	2.499341286	10.0.2.15	10.227.107.1…	TCP	54	58840 → 443 [ACK] Seq=518 Ack=3168 Win=36920 Len=0
50	2.500557902	10.0.2.15	10.227.107.1…	TLSv1.2	147	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
51	2.500782452	10.227.107.1…	10.0.2.15	TCP	60	443 → 58840 [ACK] Seq=3168 Ack=611 Win=65535 Len=0
52	2.502166362	10.0.2.15	10.227.107.1…	TLSv1.2	85	Encrypted Alert
53	2.502211563	10.227.107.1…	10.0.2.15	TLSv1.2	328	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
54	2.502286884	10.0.2.15	10.227.107.1…	TCP	54	58840 → 443 [RST, ACK] Seq=642 Ack=3442 Win=39760 Len=0
55	2.502292632	10.227.107.1…	10.0.2.15	TCP	60	443 → 58840 [ACK] Seq=3442 Ack=642 Win=65535 Len=0
56	2.502313186	10.0.2.15	10.227.107.1…	TCP	54	58840 → 443 [RST] Seq=642 Win=0 Len=0
57	2.502439132	10.227.107.1…	10.0.2.15	TCP	60	443 → 58840 [RST, ACK] Seq=3228855295 Ack=642 Win=0 Len=0
84	7.488749382	10.227.107.1…	10.0.2.15	TCP	60	80 → 58374 [FIN, ACK] Seq=572 Ack=415 Win=65535 Len=0
85	7.488887055	10.0.2.15	10.227.107.1…	TCP	54	58374 → 80 [FIN, ACK] Seq=415 Ack=573 Win=30263 Len=0
86	7.488989508	10.227.107.1…	10.0.2.15	TCP	60	80 → 58374 [ACK] Seq=573 Ack=416 Win=65535 Len=0

Figure B.1: Scenario 1 Wireshark logs.

B.3.3 Experimental scenario 3

B.3.3.1 Setup

1. **Add new authorized entity script**

   Usage of this script can be reviewed in Annex A.3.1.

   Specific command and output to authorize entities that belong to "FEUP" Organization and have Common Name "tese.client" was:

```
> sudo python add_new_accepted_entity.py -o FEUP -cn tese.client

Successfully added:
config_file : /etc/apache2/valid_expressions.list
```

```
country : None
organization_unit : None
state : None
location : None
common_name : tese.client
organization : FEUP

 Please restart apache server to apply these changes.
```

2. **Authorized entities list**

   Information below located at '/etc/apache2/valid_expressions.list':

```
SSLRequire ( %{SSL_CLIENT_S_DN_O}  eq "NOTFEUP" \
and %{SSL_CLIENT_S_DN_CN} in {"tese.negative", "CaaA", "Deaav"}) \
or (%{SSL_CLIENT_S_DN_ST} eq "PORTO" and %{SSL_CLIENT_S_DN_CN} in
{"tese.asdfe", "CaaA"}) \
or (%{SSL_CLIENT_S_DN_O} eq "FEUP" \
and %{SSL_CLIENT_S_DN_CN} eq "tese.client")
```

   It is visible that the last line (after the last 'or' statement) tries to match SSL client certificate's distinguished name organization (DN_O) with the value "FEUP" and its distinguished name common name (DN_CN) with "tese.client", the result from the execution of the commands in step 1 above.

3. **Client certificate**

   Client certificate follows the same format approximately of all other digital certificates. Here are only displayed the command to verify the information in the certificate. Data in the certificate not related to the subject that owns or signed the certificate has been truncated:

```
> openssl x509 -in client_cert.pem -text

Certificate:
    <truncated information>
        Issuer: C=PT, ST=PORTO, L=PORTO, O=FEUP,
        CN=gp-pc.tese.local/emailAddress=up201302828@fe.up.pt
        Validity
            Not Before: Apr  9 11:48:24 2018 GMT
```

```
    Not After : Apr  9 11:48:24 2019 GMT
Subject: C=PT, ST=PORTO, O=FEUP,
CN=tese.client/emailAddress=up201302828@fe.up.pt

<truncated information>
```

B.3.3.2 Network packet analysis

Figure B.2 shows the network packets transmitted between *tuxSV1* (192.168.101.172) and *tuxCLI* (10.0.2.15), on a browser access with TLSv1.2 and mutual authentication.

No.	Time	Source	Destination	Protocol	Length	Info
17	0.0190464…	10.0.2.15	192.168.101.172	TLSv1.2	198	Client Hello
19	0.0209561…	192.168.101.172	10.0.2.15	TLSv1.2	1514	Server Hello
21	0.0210045…	192.168.101.172	10.0.2.15	TLSv1.2	1395	Certificate, Server Hello Done
23	0.0220206…	10.0.2.15	192.168.101.172	TLSv1.2	396	Client Key Exchange, Change Cipher Spec, Finished
25	0.0273563…	192.168.101.172	10.0.2.15	TLSv1.2	352	New Session Ticket, Change Cipher Spec, Finished
26	0.0276248…	10.0.2.15	192.168.101.172	HTTP	443	GET / HTTP/1.1
28	0.0284188…	192.168.101.172	10.0.2.15	TLSv1.2	107	Hello Request
29	0.0285212…	10.0.2.15	192.168.101.172	TLSv1.2	235	Client Hello
31	0.0293977…	192.168.101.172	10.0.2.15	TLSv1.2	1514	Server Hello
33	0.0294999…	192.168.101.172	10.0.2.15	TLSv1.2	1729	Certificate, Certificate Request, Server Hello Done
35	0.7394278…	10.0.2.15	192.168.101.172	TLSv1.2	2005	Certificate, Client Key Exchange, Certificate Verify,
38	0.7827619…	192.168.101.172	10.0.2.15	TLSv1.2	1669	New Session Ticket, Change Cipher Spec, Finished
40	0.7835716…	192.168.101.172	10.0.2.15	TLSv1.2	1514	[SSL segment of a reassembled PDU]
44	0.7836333…	192.168.101.172	10.0.2.15	TLSv1.2	850	HTTP/1.1 200 OK (text/html)
45	0.8183061…	10.0.2.15	192.168.101.172	HTTP	395	GET /icons/openlogo-75.png HTTP/1.1
47	0.8189498…	192.168.101.172	10.0.2.15	TLSv1.2	107	Hello Request
48	0.8190727…	10.0.2.15	192.168.101.172	TLSv1.2	235	Client Hello
50	0.8204192…	192.168.101.172	10.0.2.15	TLSv1.2	3013	Server Hello, Certificate, Server Hello Done
54	0.8219214…	10.0.2.15	192.168.101.172	TLSv1.2	485	Client Key Exchange, Change Cipher Spec, Finished
56	0.8265435…	192.168.101.172	10.0.2.15	TLSv1.2	405	New Session Ticket, Change Cipher Spec, Finished
57	0.8267015…	192.168.101.172	10.0.2.15	TLSv1.2	2974	[SSL segment of a reassembled PDU]
61	0.8268462…	192.168.101.172	10.0.2.15	HTTP	1812	HTTP/1.1 200 OK (PNG)
92	5.8298972…	10.0.2.15	192.168.101.172	TLSv1.2	107	Alert (Level: Warning, Description: Close Notify)

Figure B.2: Scenario 3 Wireshark logs.

After the standard TLS handshake opening messages, the server requests client's certficate on packet number 33 ("Certificate Request").

Soon after the client replies with its certificate, TLS handshake process is completed and HTML data and even images present in *tuxSV1*'s server landing page are successfully transmitted.

B.3.4 Experimental scenario 4

Network packets in figure B.3 follow a pattern similar to scenario 3 until packet number 42, where *tuxSV1* sends a Forbidden message (denying *tuxCLI* access). It is noteworthy that this happens right after packet number 39 with the client's certificate is sent, so a confident assumption that verification and subsequent rejection of the client's certificate is what caused that access denial.

No.	Time	Source	Destination	Protocol	Length	Info
21	0.007857672	10.0.2.15	192.168.101.172	TLSv1.2	198	Client Hello
23	0.008970852	192.168.101.172	10.0.2.15	TLSv1.2	1514	Server Hello
25	0.009048009	192.168.101.172	10.0.2.15	TLSv1.2	1395	Certificate, Server Hello Done
27	0.009945270	10.0.2.15	192.168.101.172	TLSv1.2	396	Client Key Exchange, Change Cipher Spec, Finished
29	0.014162353	192.168.101.172	10.0.2.15	TLSv1.2	352	New Session Ticket, Change Cipher Spec, Finished
30	0.014527761	10.0.2.15	192.168.101.172	HTTP	443	GET / HTTP/1.1
32	0.015260959	192.168.101.172	10.0.2.15	TLSv1.2	107	Hello Request
33	0.015357374	10.0.2.15	192.168.101.172	TLSv1.2	235	Client Hello
35	0.016240538	192.168.101.172	10.0.2.15	TLSv1.2	1514	Server Hello
37	0.016264388	192.168.101.172	10.0.2.15	TLSv1.2	1729	Certificate, Certificate Request, Server Hello Done
39	0.443282787	10.0.2.15	192.168.101.172	TLSv1.2	2005	Certificate, Client Key Exchange, Certificate Verify,
42	0.493220944	192.168.101.172	10.0.2.15	HTTP	2143	HTTP/1.1 403 Forbidden (text/html)
44	2.143097599	10.0.2.15	192.168.101.172	TLSv1.2	107	Alert (Level: Warning, Description: Close Notify)

Figure B.3: Scenario 4 Wireshark logs.

B.3.5 Data Mining process - model transmission

Revealed on figure B.4, standard handshake is performed on packets 641 to 662.

The Data Mining model being transferred was sent over packet number 664, using TLSv1.2 as well.

No.	Time	Source	Destination	Protocol	Length	Info
641	45.607722596	10.0.2.15	10.227.107.147	TLSv1.2	1849	Client Hello
644	45.612636621	10.227.107.147	10.0.2.15	TLSv1.2	2894	Server Hello
646	45.612740367	10.227.107.147	10.0.2.15	TLSv1.2	381	Certificate, Server Key Exchange, Server Hello Done
648	45.613636217	10.0.2.15	10.227.107.147	TLSv1.2	147	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
650	45.613750229	10.0.2.15	10.227.107.147	TLSv1.2	505	Application Data
652	45.615389805	10.227.107.147	10.0.2.15	TLSv1.2	328	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
653	45.616039964	10.227.107.147	10.0.2.15	TLSv1.2	87	Encrypted Handshake Message
655	45.616316744	10.0.2.15	10.227.107.147	TLSv1.2	290	Encrypted Handshake Message
657	45.620460104	10.227.107.147	10.0.2.15	TLSv1.2	3525	Encrypted Handshake Message, Encrypted Handshake Message, Encrypted Han
659	45.625896638	10.0.2.15	10.227.107.147	TLSv1.2	1715	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Me
662	45.669875809	10.227.107.147	10.0.2.15	TLSv1.2	1608	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Me
664	45.670149997	10.227.107.147	10.0.2.15	TLSv1.2	1446	Application Data
674	50.670534401	10.0.2.15	10.227.107.147	TLSv1.2	85	Encrypted Alert
678	50.671443595	10.227.107.147	10.0.2.15	TLSv1.2	85	Encrypted Alert

Figure B.4: Secure Data Mining model transmission

Data carried in packet 664 is unintelligible, as seen in figure B.5.

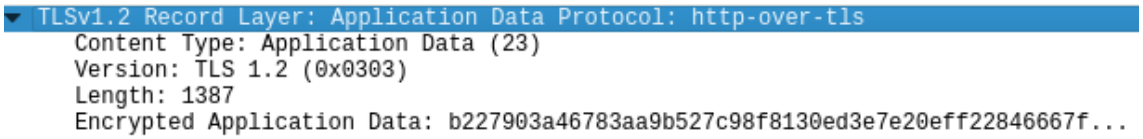


Figure B.5: Packet 664: Encrypted application data

Appendix C

Data Mining Processes

C.1 File server status after workflow

C.1.1 Datasets

Index of /datasets

Name	Last modified	Size	Description
Parent Directory		-	
adult.test/	2018-06-26 00:36	-	
adult2/	2018-06-26 00:23	-	
adult3/	2018-06-26 00:26	-	
anon2/	2018-06-26 02:48	-	
anon3/	2018-06-26 02:48	-	
generalized_all_16.test/	2018-06-26 02:50	-	

Apache/2.4.29 (Ubuntu) Server at douro.fe.up.pt Port 443

Figure C.1: Post Data Mining application: shared datasets on douro machine

C.1.2 Models

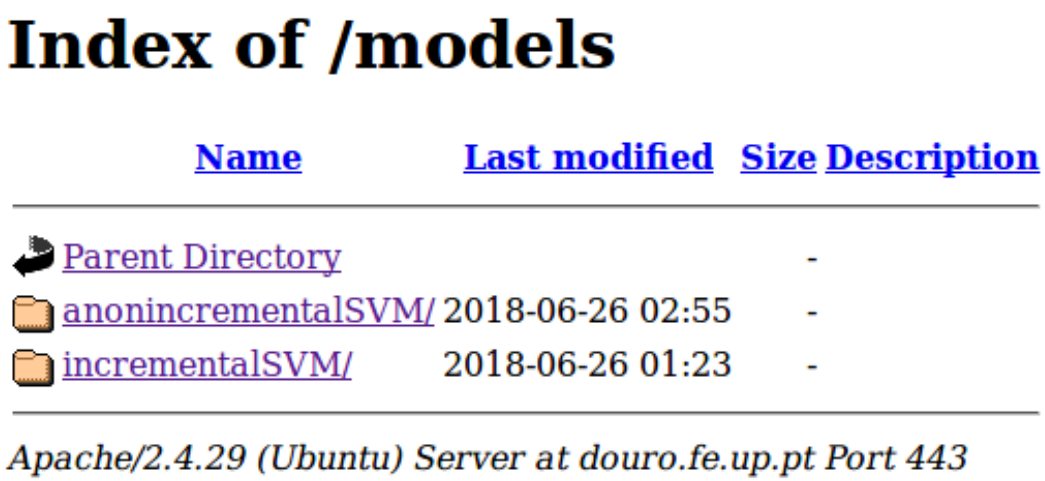


Figure C.2: Post Data Mining application: shared models on douro machine

C.2 Intermediate incremental results

C.2.1 Original dataset

Figures C.3 and C.4.

```
[[6647 1549]
 [2184  366]]
Mean accuracy: 0.335546249767
Maximum accuracy: 0.652614926484
Minimum accuracy: 0.241299088033
```

Figure C.3: Original adult dataset: *tux1* intermediate increment.

C.2.2 Anonymized dataset

Figures C.5 and C.6.

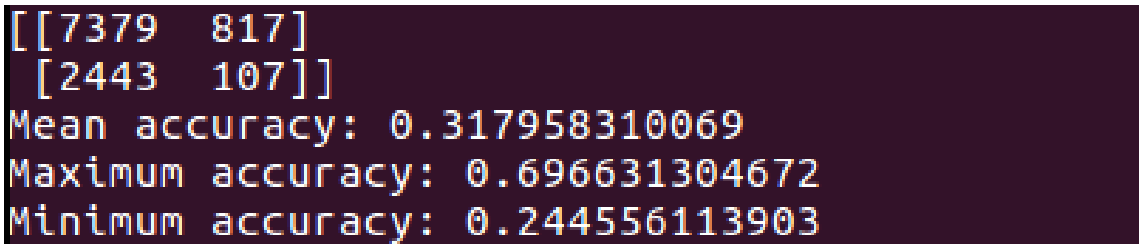


Figure C.4: Original adult dataset: *tux2* intermediate increment.

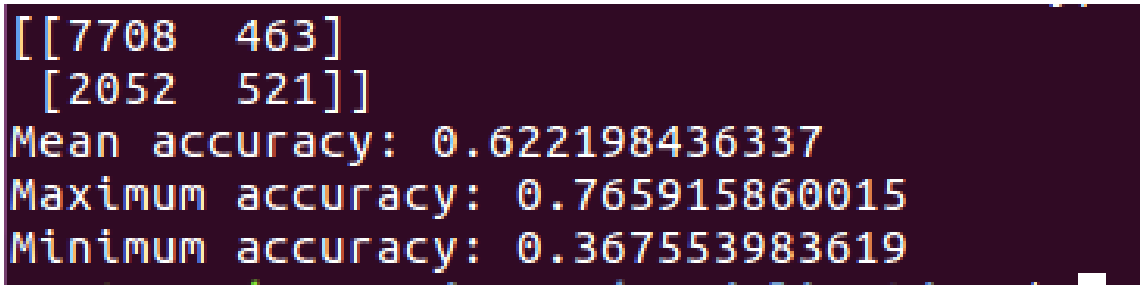


Figure C.5: Anonymized adult dataset: *tux1* intermediate increment.

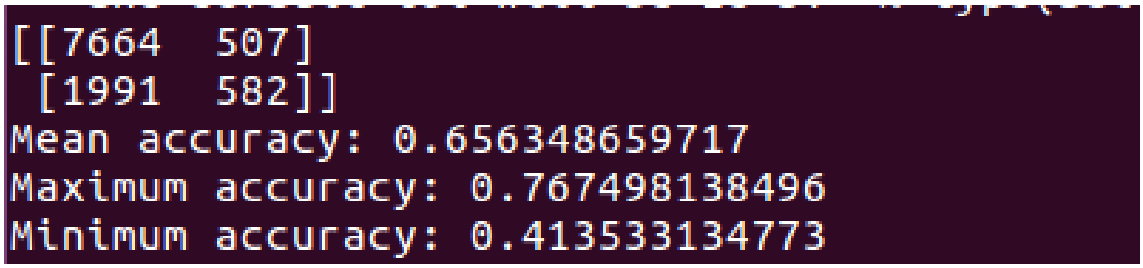


Figure C.6: Anonymized adult dataset: *tux2* intermediate increment.

C.3 Online Random Forest

C.3.1 Dataset description

Existing attributes are:

```
    @ATTRIBUTE id NUMERIC
@ATTRIBUTE subjectID NUMERIC
@ATTRIBUTE windowID NUMERIC
@ATTRIBUTE mean_x NUMERIC
@ATTRIBUTE mean_y NUMERIC
@ATTRIBUTE mean_z NUMERIC
@ATTRIBUTE sd_x NUMERIC
@ATTRIBUTE sd_y NUMERIC
@ATTRIBUTE sd_z NUMERIC
```

```
@ATTRIBUTE variance_x NUMERIC
@ATTRIBUTE variance_y NUMERIC
@ATTRIBUTE variance_z NUMERIC
@ATTRIBUTE median_x NUMERIC
@ATTRIBUTE median_y NUMERIC
@ATTRIBUTE median_z NUMERIC
@ATTRIBUTE iqr_x NUMERIC
@ATTRIBUTE iqr_y NUMERIC
@ATTRIBUTE iqr_z NUMERIC
@ATTRIBUTE mad_x NUMERIC
@ATTRIBUTE mad_y NUMERIC
@ATTRIBUTE mad_z NUMERIC
@ATTRIBUTE kurtosis_x NUMERIC
@ATTRIBUTE kurtosis_y NUMERIC
@ATTRIBUTE kurtosis_z NUMERIC
@ATTRIBUTE energy_x NUMERIC
@ATTRIBUTE energy_y NUMERIC
@ATTRIBUTE energy_z NUMERIC
@ATTRIBUTE entropyF_x NUMERIC
@ATTRIBUTE entropyF_y NUMERIC
@ATTRIBUTE entropyF_z NUMERIC
@ATTRIBUTE meanDC_x NUMERIC
@ATTRIBUTE meanDC_y NUMERIC
@ATTRIBUTE meanDC_z NUMERIC
@ATTRIBUTE cc_x NUMERIC
@ATTRIBUTE cc_y NUMERIC
@ATTRIBUTE cc_z NUMERIC
@ATTRIBUTE gravity_x NUMERIC
@ATTRIBUTE gravity_y NUMERIC
@ATTRIBUTE gravity_z NUMERIC
@ATTRIBUTE orientation_x NUMERIC
@ATTRIBUTE orientation_y NUMERIC
@ATTRIBUTE orientation_z NUMERIC
@ATTRIBUTE entropyT_x NUMERIC
@ATTRIBUTE entropyT_y NUMERIC
@ATTRIBUTE entropyT_z NUMERIC
@ATTRIBUTE orientationDZ NUMERIC
@ATTRIBUTE humanactivities STRING
@ATTRIBUTE humanpositions STRING
@ATTRIBUTE deviceposition STRING
```

Possible labels or classes are:

```
@ATTRIBUTE class {climbingdown,climbingup,jumping,lying,
standing,sitting,running,walking}
```

C.3.2 Intermediate distributed result

Results presented in figure C.7.

```
Serialized model is saved in /tmp/serialized.ser
Correctly Classified Instances        6247               72.7410 %
Incorrectly Classified Instances      2341               27.2590 %
Total Number of Instances             8588

FP Rate   Precision   Recall   F-Measure   Class
  0.016       0.884    0.959       0.920   a = climbingdown
  0.027       0.854    0.922       0.887   b = climbingup
  0.073       0.195    1.000       0.327   c = jumping
  0.000       ◆    0.000       ◆   d = lying
  0.007       0.958    0.984       0.971   e = standing
  0.167       0.510    0.992       0.673   f = sitting
  0.000       1.000    0.266       0.421   g = running
  0.017       0.903    0.941       0.922   h = walking
---------------------------------------------------
  0.036       ◆    0.727       ◆   Weighted Avg.

    a     b    c  d     e     f    g     h   <-- classified as
  947     8    0  0    17     0    0    16 |   a = climbingdown
    0  1166    0  0    27     0    0    71 |   b = climbingup
    0     0  150  0     0     0    0     0 |   c = jumping
    2     3    0  0     5  1210    0     0 |   d = lying
    2     3    0  0  1228     7    0     8 |   e = standing
    2     3    0  0     3  1269    0     2 |   f = sitting
  118   113  618  0     0     2  319    28 |   g = running
    0    69    0  0     2     2    0  1168 |   h = walking
```

Figure C.7: Final distributed ORF results